コードクローンの作者に関する定量的分析

2019SE023 小島愛由　2019SE052 佐藤 優里愛
指導教員：横森励士，井上克郎

1　はじめに

コードクローン（以降クローン）とは，同一または類似したソースコードのことであり，主にコピー＆ペーストによって作成される．互いにクローンの関係にあるソースコードをクローン片（もしくは単にクローン）と呼び，クローン対をクローンペアと言う．クローン関係にあるクローンの同値類をクローンセットと呼ぶ [1].

ソースコードのコピー＆ペーストによる機能の実現は，似た内容の処理を実現するための簡便な手法としては有用であるが，後にクローンとなっている部分に修正が必要となった場合，その修正をクローンセットの各ソースコードに反映させる必要がある．大規模なソースコードにクローンが大量に分布している場合，１つの修正に対して膨大な修正が必要となる可能性があり，保守作業に多大なる影響を及ぼす.

近年クローンの分析やその保守についての多くの研究論文が発表され，また，実用的なクローン検出ツールの販売もされている [2]．しかし，クローンの特性，特にクローンを形成するコードの作者について，詳しい分析はあまりされていない.

本研究では，GitHub 上で公開されている規模や参加者数の異なるいろいろなプロジェクトに対して，クローンの作者を分析し，プロジェクトによるクローンの作者の違いを明らかにする．また，クローンの作者の相違により，クローンの特性に違いがあるのかを調べる.

3 つのプロジェクトの分析として，(1) クローンセットの総数とクローンの行数の値や要素数の値などの基本的な特性はプロジェクトによって異なる，(2) プロジェクト全体でソースコード作成に貢献していない作者が，多くのコードクローンを作成することがある，(3) 全てのプロジェクトでは，非単一クローンの方が単一クローンよりも行数が大きい，などが分かった.

本研究のクローンと作者の関係を抽出する仕組みは，クローン情報にその作者を表示する機能，異なる作者のクローンを編集する際に警告を表示する機能などに活用できると考える．また，本研究の成果は，クローンに関係するバグの予測や保守作業に役立つことが期待される.

2　既存研究

開発に携わる人数とクローンが変更される確率の間の影響を調査するために，Harder は [3] で，クローンとなっているソースコードを誰が編集していたかを追跡する方法を提案した（以降，この編集者のことを作者と呼ぶ）．調査の結果，66.3 %のクローンは 1 人の作者によって作成されたことが分かった．一方で 33.7 %のクローンは少なくとも 2 人の異なる作者によって作成されたことが分かった．また，2 人の異なる作者によって作成されたクローンセットが変更される確率は，1 人の作者によって作成されたクローンセットの変更される確率の 2 倍となることを示した．この研究はひとつのプロジェクトを対象として分析した結果であり，他のプロジェクトでの分析結果は示されていない．本研究では複数のプロジェクトの分析を行い，特性の違いを調べた.

2.1　コードクローンに関する用語と前提

本研究では，ある行を最後に編集しコミットした人を行の作者とする．あるクローンで一番多くの行の作者をそのクローンの全体の作者とし，クローンセットで一番多くのクローンの作者をクローンセット全体の作者とする．また，クローンセットにおいて，含まれるクローンの作者が全て同一である場合，単一クローンセット，含まれるクローンの作者が複数人である場合，非単一クローンセットと呼ぶ.

この分析では，クローンペアではなく，クローンセットに基づいて分析を行う．クローンペアでの解析は単純で解析しやすいが，要素数が大きなクローンセットに対しては多くのクローンペアが存在し，定量的な評価がわかりにくくなる．これは Harder[3] の先行研究と同じである.

3　リサーチクエスチョン

本研究では，複数のプロジェクトのクローンの作者の特性を調べるために以下の 3 つのリサーチクエスチョンを設けた.

RQ1：対象プロジェクト中の全クローンの基本的な特性は何か

三つのプロジェクトにクローンセットの総数やその要素数，クローンの行数などの基本的な特性に違いがあるかを調査した.

RQ2：クローンセットの作者の単一性がプロジェクトによってどのように異なるか

各プロジェクトのクローンの作者にどのような違いがあるかを調査した.

RQ3：単一クローン対と非単一クローン対には，どのような違いがあるのか

一人の作者によるクローンと複数作者によるものでは，何か特性の違いがあるか調査した.

4　分析対象プロジェクトの概要

分析対象プログラムとして，GitHub で公開されている Mindustry[*1]，Tomcat[*2]，disruptor[*3]の三つを選択した．Mindustry は，ベルトコンベアで緻密なサプライチェーンを作成することが目的のゲームである．Tomcat は，Java Servlet や JavaServer Pages(JSP) を実行するための Web コンテナである．disruptor は，スレッド間のメッセージ交換を高速に行うライブラリである．

これらの対象プロジェクトの概要を表 1 に記した．Mindustry と Tomcat は比較的大規模なプロジェクトで，作者数も多い．disruptor は比較的小さなソフトウェアであるが，作者数は 50 人を超えており，多数の人間が開発に関わっている．

5　分析方法

コードクローンやその作者，特性の分析を以下の手順で行った (図 1).

1. 分析対象のプロジェクトを GitHub より入手する．
2. CCFinderSW[4] を実行して, 分析対象のクローンセットの情報や構成ファイルの情報を入手する．最小検出トークン数 t は 50, 非繰り返し割合 rnr は 0.3 とした．
3. 得られた各クローンの行に対して git blame を実行するためのコマンドを生成する．
4. git blame を実行して各行を誰が最後に作成したかの情報を取得する．
5. 出力結果をクローンセットごとにまとめる．
6. 各作者の出現回数を求めるプログラムを実行する．
7. 各クローンセットのクローンごとの作者を求めるプログラムを実行する．
8. インスタンスごとの作者から，クローンセットの作者を求めるプログラムを実行する．

6　分析結果

6.1　RQ1

クローンセットの総数を表 2 に，クローンの行数の概要を表 3 に示す．クローンの行数の分布（ヒストグラム）は、例えば Tomcat の場合は図 2 のようになった．クローンの行数の分布は 1〜29 行のものが 1 番多く，行数が増えるにつれクローンの個数は減っている．クローンの行数が 200 行以下のものが 9 割以上を占めている．Mindustry もほぼ同様な傾向であるが，disruptor では，30〜59 行のものが最多となっており，比較的長いクローンが多いことがわかる．

クローンセットの要素数の概要は表 4 のようになり，分布（ヒストグラム）は，Tomcat の場合は図 3 のようになり，2 個のものが 1 番多かった．要素数が増えるにつれ，要素は減っている．要素数が 2 個のものが 5 割以上を占めている．これは Mindustry や disruptor もほぼ同様な傾向である．

RQ1 の結論：クローンセットの総数とクローンの行数の値や要素数の値などの基本的な特性はプロジェクトによって異なっていた．しかし，クローンセットの要素数に関しては，どのプロジェクトも最小値と中央値が供に 2 であり，複数回コピー＆ペーストされたクローンは少なかったと考えられる．

6.2　RQ2

各プロジェクトごとに，クローンセット全体をまとめて解析対象として，作者に関する情報を求めた．以下，Tomcat を中心に述べる．

作者の分布は図 4 のようになった．プロジェクト全体の 60 %が Mark により作成されていたのに対して（上図），クローン各行の 53 %が Mark により作成されていた（下図）．また，プロジェクト全体の 5 %を作成していた remm は，クローン各行の 17 %を作成していた．

クローンセット中の各クローンの作者数の概要は表 5 のようになり，分布は図 5 のようになった．

プロジェクト全体の約 46 %を単一作者が作成しており，単一クローンセットの割合が非単一クローンセットの割合よりも小さかった．また，関与している作者数が増えるにつれ，クローンセットの数は減少している．Mindustry では約 61 %が単一作者が作成していた．disruptor では，約 24 %が単一作者のクローンであった．

RQ2 の結論：プロジェクト全体でソースコード作成に貢献していない作者が，多くのコードクローンを作成することがあった．また，単一作者のクローンの割合は 24 %〜62 %であった．

6.3　RQ3

単一クローンセットと非単一クローンセットの特性の違いを Tomcat について述べる．

Tomcat の単一クローンセットの行数は，最大値が 84 行，最小値が 1 行，中央値が 24 行，平均値が約 44.2 行となり，四分位範囲は 28 となった．また，非単一クローンセットの行数は，最大値が 113 行，最小値が 2 行，中央値が 31 行，平均値が約 47.7 行となり，四分位範囲は 38 となった．このように Tomcat では中央値や平均値は，非単一クローンの方が単一クローンより大きく，大きなクローンが多いことが分かる．他の 2 つのプロジェクトも同様の傾向を示した．

Tomcat の単一クローンセットの要素数は，最大値が 7 個，最小値が 1 個（2 行以下のクローンを削除したため），平均値が約 5.2 個となり，四分位範囲は 2 となった．また，非単一クローンセットの要素数は，最大値が 7 個，最小値が 2 個，平均値が約 3.7 個となり，四分位範囲は 2 となっ

[*1] https://github.com/Anuken/Mindustry
[*2] https://github.com/apache/tomcat
[*3] https://github.com/LMAX-Exchange/disruptor

表 1　対象プロジェクトの概要

プロジェクト	レポジトリ	コミッター	総行数	ファイル数
Mindustry	1.73GB	491 人	142,647 行	740 ファイル
Tomcat	142.4MB	93 人	622,671 行	2,615 ファイル
Disruptor	9.27MB	54 人	28,059 行	235 ファイル

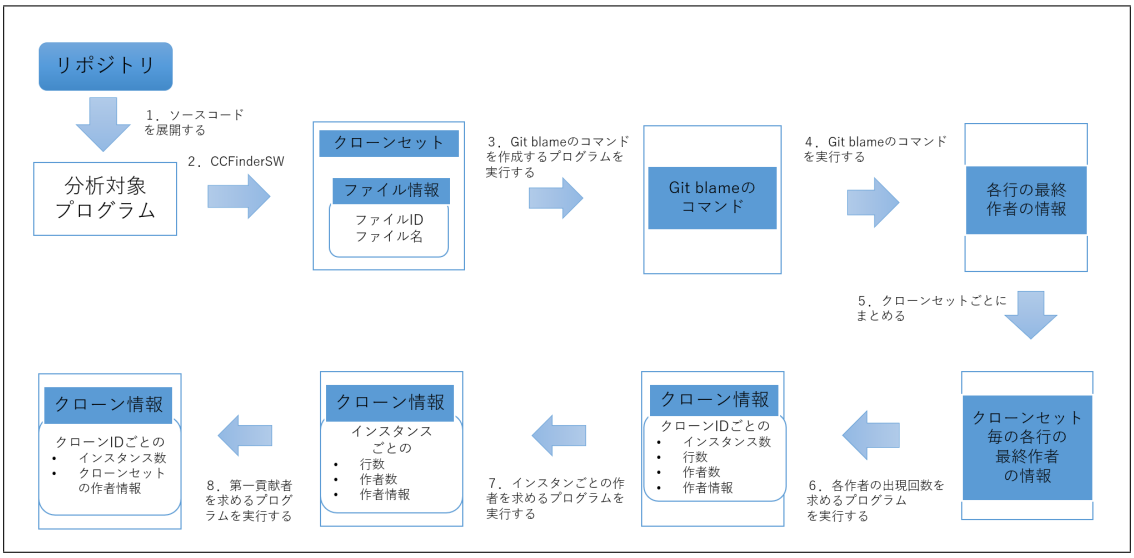

図 1　コードクローンに対する分析の手順の流れ図

表 2　クローンセットの総数

プロジェクト	クローンセットの総数	クローンの総行数
Mindustry	2,131 個	56,737 行
Tomcat	4,284 個	197,447 行
Disruptor	567 個	51,507 行

表 3　クローンの行数の概要（単位：行）

プロジェクト	最大数	最小数	中央値	平均値
Mindustry	324	2	22	26.6
Tomcat	2,338	1	27	46.1
Disruptor	3,096	1	47	90.8

表 4　クローンセットの要素数の概要（単位：個）

プロジェクト	最大数	最小数	中央値	平均値
Mindustry	50	2	2	2.8
Tomcat	334	2	2	4.4
Disruptor	35	2	2	2.6

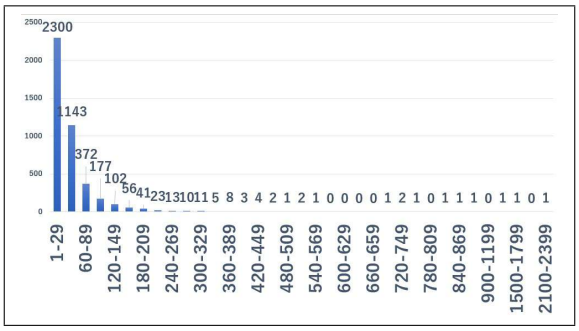

図 2　クローンの行数の分布 (Tomcat)

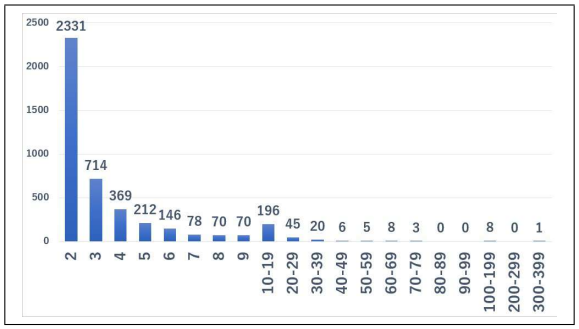

図 3　クローンセットの要素数の分布 (Tomcat)

た．平均値については，単一クローンのほうが大きいが，他の 2 つのプロジェクトでは非単一クローンの方が大きくなっており，プロジェクトによる違いがあった．

RQ3 の結論：全てのプロジェクトでは，非単一クローンの方が単一クローンよりも行数の平均値が大きかった．また，非単一クローンセットと単一クローンセットの要素数の差は，プロジェクトによって異なることが分かった．

その理由として，単一クローンは作者が同じなため自分の考えが書けるが，非単一クローンの場合は別の作者の

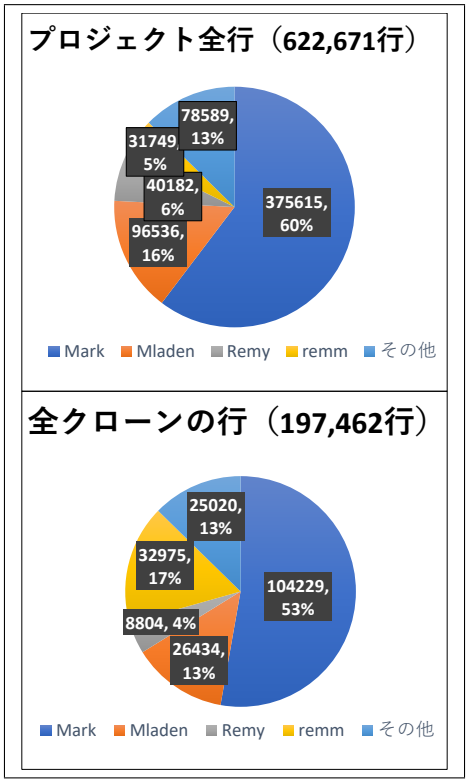

図 4　ソースコードの各行の作者の分布 (Tomcat)

表 5　クローンセット中のクローンの作者数の概要

プロジェクト	最大数	最小数	平均値	単一作者の割合
Mindustry	8 人	1 人	1.5 人	1,302 個 (61.1 %)
Tomcat	11 人	1 人	2.0 人	1,953 個 (45.6 %)
Disruptor	13 人	1 人	2.8 人	133 個 (23.5 %)

コードを利用するため，付加的な処理が必要になる場合があるためと思われる．

7　議論

7.1　考察

3 つのプロジェクトの単一作者の割合は，24 %〜62 %で，それらの平均値は 43.4 %で，Harder の結果 [3] の 66.3 %より少なかった．これは Harder の分析対象は作者数が 23 名と少ないことも関係していると思われる．

また，今回対象としたプロジェクトの Mindustry はゲーム，Tomcat は Web コンテンツ，Disruptor はスレッド間メッセージングライブラリである．プロジェクト内容の違いがクローンや作者の人数に関係している可能性がある．

7.2　妥当性の脅威

分析を簡単化するために，一番多くの行や要素を作成した作者だけに注目して分析した．そのため 2 番目以降の

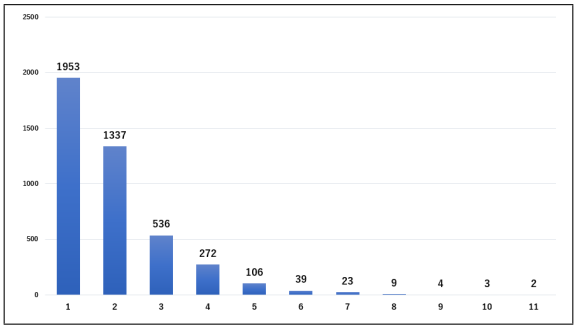

図 5　クローンセット中のクローンの作者数の分布 (Tomcat)

作者の情報が欠落する恐れがある．また，今回は 3 つのプロジェクトの分析しか行わなかったが，プロジェクトの違いを明確にするためにはより多くのプロジェクトを分析し，統計的に処理する必要ある．さらに今回はクローン検出ツールとして CCFinderSW を用いたが，他のツールでは結果が変わる恐れがある．また，CCFinderSW のパラメータにも影響される可能性がある．

8　おわりに

本研究は，「多人数のプロジェクトでは，コードに関わる人も増え，クローン作者も多数になるのではないだろうか」，また，「作者が同一のコードクローンと異なるコードクローンでは，どのような違いがあるのだろうか」，といった疑問について調査を行ってきた．

今回提案した 3 つのリサーチクエスチョンに従い，GitHub 上で公開されている規模や参加者数の異なる 3 つプロジェクトに対して，クローンの作者を分析し，クローンの作者の相違により，クローンの特性に違いがあるのかを調べることが出来た．その中で非単一クローンの方が行数が長く複雑度が高いことが分かったため，第 1 章で述べた機能に非単一クローンの時にはより強い警告を出す機能を付け加えるなどして活用できる．

今後は，多数のプロジェクトを対象としてクローンやクローン作者の特性を統計的に分析していきたい．

参考文献

[1] 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, 2008.

[2] Katsuro Inoue, Chanchal K. Roy, edited by: ”Code Clone Analysis: Research, Tools, and Practices”, Springer, ISBN 987-981-16-1926-7, Aug., 2021.

[3] Jan Harder：”Code Clone Authorship — A First Look”, Softwaretechnik Trends, 32(2), pp.25–26, 2012.

[4] CCFinderSW https://github.com/YuichiSemura/CCFinderSW.