コードクローンを互いに有する部品の配置に関する分析

2011SE201 野々山未菜
指導教員：横森励士

1　はじめに

近年，ソフトウェアは大規模化しつつあり，構成する部品も増大している．このような環境下では，メトリクスに基づいてある特性を持つ部品を抽出するといった手法を用いて，部品を効果的に管理する方法が求められている．先行研究 [1] では，コンポーネントランク [2] の計算において，コードクローン [3] を互いに有する部品を統合することで，コードクローンとなる部分から良く利用される部品を抽出する手法が提案されている．提案手法における評価実験では，部品グラフの構造を壊さないように結合を制限することで，想定した結果が得やすいことが推測できた．しかし具体的にどうすべきかについては言及されていない．

本研究では，先行研究における解析の精度を向上させるため，「部品が属する Java パッケージ間の距離」に基づいて結合を制限する方法が妥当であるかを検証する．具体的にはオープンソースのプロジェクトにおいて，利用方法が同じコードクローンを抽出し，それらが属するクラスが，パッケージ配置上でどれくらい離れているかを分析する．また，離れている部品についてはどのような部品であるかを分析する．これらの分析を行うことで，パッケージ間の距離に基づき，結合を制限する方法が先行研究の精度の向上に役立つことを確認する．

2　分析内容

本研究は，先行研究 [1] での提案手法の精度を向上させる方法として，コードクローンを持つ部品について，パッケージ間の距離を元に結合の可否の判断を行うことが妥当であるかを検証することを目的とする．ここでいう部品とは，クラスなどのソフトウェアの構成要素である．

2.1　分析の手順

1. Java プログラムを CCFinder[3] と Classycle[4] に入力し，クローン関係と利用関係を取得する．
2. クローン関係を持つクラス同士を結合し，クラス群を作成し，プロジェクト毎に抽出する．
3. 抽出したクラス群それぞれについて，コードクローンの中を調査し，群内の他のクラスでも同じような方法で他のクラスを利用しているかを調べる．このような事例が 1 つでも存在するクラス群のみを残し，解析対象とする．

2.2　分析の項目について

これらのクラス群に対して，次のような項目を調査した．

1. それぞれのパッケージ内のクラスについて 2 つのクラス間の距離の最小，最大，平均を分析する．ここでいう距離とは，パッケージの階層をつたって移動したときに，もう一方のパッケージに到達するまでの移動回数である．実際のプロジェクトにおいてコードクローンを持つ部品同士がどれだけ離れているかが分かる．よって，部品同士の結合を制限する判断材料となる．
2. 次に，パッケージ間の距離が離れたクラス群には，どのようなクラスが含まれているかを分析する．これによって，離れたパッケージ間に存在するコードクローンの傾向をつかむことができる．また，実際のプロジェクトでどのような意図をもって部品が配置されているかを理解できる．

3　パッケージ間の距離に関する分析

オープンソースプロジェクト 35 個に対して分析を行い，53 のクラス群を抽出した．プロジェクト毎のクラス群の抽出状況を表 1 に示す．表 1 からはクラス群数が 1 以下のプロジェクトは 23 と多い一方で，多くのクラス群を持つプロジェクトも少なからず存在するということが分かる．

表 1　各プロジェクトにおけるクラス群の抽出状況

存在するクラス群数	0	1	2	3	4	11
プロジェクト数	12	11	6	1	4	1

53 のクラス群中に存在するクラスの総数は 217 個であり，1 つのクラス群には平均して約 4.09 個のクラスが存在した．分布状況を表 2 に示す．表 2 からは，クラス数が 3 以下のクラス群数は 36 と多い一方で，たくさんのクラスが結合してできるクラス群も存在することが分かる．

表 2　各クラス群内のクラス数

群内のクラス数	2	3	4	5	6	7	9	10〜
クラス群数	24	12	3	3	5	1	1	4

それぞれのクラス群中の 2 つのクラスを選んだ時の，そのクラス間の距離の最小，最大，平均の値の分布を表 3,4,5 にそれぞれ示す．これらの表からは，約 64%(34/53) のクラス群はすべて同一パッケージ内に存在すること，また約 81%(43/53) のクラス群は，同一パッケージ内にあるクラスの組み合わせが存在することが分かる．

表 3　最小値

パッケージ間距離	0	1	2	3	4
クラス群数	43	2	4	2	2

表 4 最大値

パッケージ間距離	0	1	2	3	4	5	7
クラス群数	34	1	10	1	5	1	1

表 5 平均値

パッケージ間距離	0〜1	1〜2	2〜3	3〜4	4〜5
クラス群数	35	7	6	2	3

パッケージ間の距離の最小値が 0 の 43 個のクラス群は，同一パッケージのみ結合可という制限下でも何らかの形で結合がなされる．一方，残りの 10 のクラス群はすべてのクラスが異なるパッケージに属するので，パッケージ間の距離で結合を制限すると結合されない可能性がある．これらのクラスについて分析を行い，具体的にどのような部分がコードクローンになっているか調査する．

4 離れた部品に関する分析

前節で述べた，すべてのクラスが異なるパッケージに属する 10 のクラス群について，コードクローンの中身を調査した．以下，それぞれを大まかに分け，説明する．

- スタッツの表示など，中身が同じメソッドが存在することで発生したコードクローン．この種のクラス群のパッケージ間の距離は 1 などであった．
- 同じクラスから継承を行っていることで同一の処理記述が存在し発生したコードクローン．この種のクラス群のパッケージ間の距離は 2 などであった．
- 異なる種類の実装を実現することを目的としてクラスが構造的に一致していることで発生したコードクローン．この種のクラス群のパッケージ間の距離は 2 などであった．
  - 例：Barbecue[*1]というバーコード作成アプリケーションでは，異なる種類のバーコードを扱うクラスがバーコードの種類ごとに分類されて整理されている．バーコードの有効性を確認するためのメソッド間や，バーコードのモジュール定義において文字に対応した情報を与えるための記述のかたまりがコードクローンとして抽出できた．
- 直接的な関係が薄かったり，大まかなロジックが一致しただけのコードクローン．この種のクラス群のパッケージ間の距離は 3 などであった．
- クラスの役割が似たものであったため，役割を果たすための記述の部分で発生したコードクローン．この種のクラス群のパッケージ間の距離は 3 などであった．
  - 例：Frinika[*2]という音楽作成アプリケーションでは，外部機器と音楽データをやりとりするためのラッパークラスが複数存在し，ラッパークラスとして機能する部分にコードクローンが存在した．
- 本体の部品とそれを利用して作成した部品の間に発生したコードクローン．この種のクラス群のパッケージ間の距離は 4 などであった．
  - 例：JID[*3]という画像ダウンロードアプリケーションでは，Downloader に関する機能を扱うパッケージのクラスを継承して，test パッケージに似たような処理メソッドを持つクラスが存在する．このようにコードクローンが抽出できた．

これらより，パッケージ間の距離が小さいほどコードクローンとして考慮すべき度合いが高い傾向があると考えられる．この結果より，パッケージ間の距離による結合の条件を厳しくすると，グラフ構造を大きく変化させるような結合が減り，精度が向上するという結果が得られた [5]．

5 おわりに

本研究では，先行研究 [1] における解析の精度を向上させるため，部品が属するパッケージ間の距離に基づいて，結合を制限する方法が妥当であるかを検証した．結果として，パッケージ間の距離が離れたコードクローンは直接的な関係性が低いことが多いと分かった．また，パッケージ間の距離に基づいて結合の制限を行う方法はある程度妥当性があることを確認した．

今後の課題は，新規部品が配置されるべきパッケージを推測する手法を提案することである．

参考文献

[1] 千賀英佑：”コードクローンを利用したソフトウェア部品の評価手法についての考察”，南山大学大学院数理情報研究科 2013 年度修士論文.

[2] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, and Shinji Kusumoto, ”Ranking Significance of Software Components Based on Use Relations,” Transaction on Software Engineering, vol. 31, no. 3, pp. 213-225, 2005.

[3] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue, ”CCFinder: A Multi-Linguistic Tokenbased Code Clone Detection System for Large Scale Source Code,” Transaction on Software Engineering, vol. 28, no. 7, pp. 654-670, 2002.

[4] Classycle: Analysing Tools for Java Class and Package Dependencies, http://classycle.sourceforge.net/

[5] 奥田黎哉，小林登夢，村瀬慶紀：”コードクローンを用いたコンポーネントランク拡張手法の有効性評価 —部品グラフの変化についての分析 —”，南山大学情報理工学部 2014 年度卒業論文.

[*1] Barbecue, http://sourceforge.net/projects/barbecue/

[*2] Frinika, http://sourceforge.net/projects/frinika/

[*3] JID, http://sourceforge.net/projects/jid/