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改良を目的とした評価実験

── パッケージ間距離に基づく制限による影響の調査──

2011SE030 原田年祐　2011SE215 大野航

指導教員：横森励士

1　はじめに

近年，ソフトウェアの規模が大きくなるにつれて，ソフトウェアを構成する部品を効果的に管理することが求められている．千賀らは，コンポーネントランクの計算においてコードクローン関係を考慮することで，コードクローンを持つ部品から共通して利用されている部品を検出する方法を提案した [4]．評価実験 [5]，[6] では，手法の有効性を確認したが，実際の利用のためにはさらなる精度の向上をはかる必要であると考えた．本研究では，所属するパッケージがどれだけ離れているかの情報に基づいて，一定以上離れた部品間に存在するコードクローンを考慮しないことで，手法の精度が向上するかを確認する．離れたパッケージに属するクラス間に存在するコードクローンは関係の弱いものが多いという仮説をたて，部品グラフの構造が想定以上に変化するようなノード結合が起こらないようにすることで，精度が向上すると考えた．実際にオープンソースの開発プロジェクトに対して適用を行い，提案する手法が精度の向上に役立つかどうかを検証する．[5] の手法が，十分な精度を持つようになることで，ソフトウェアの保守作業の支援に役立てることができると考えられる．

2　背景技術

2.1　部品グラフとコンポーネントランク

一般に，ソフトウェア部品とはモジュールや，クラスなどのソフトウェアの構成要素を指す．以降，ソフトウェア部品を単に部品と呼ぶ．ソフトウェアは構成要素の部品間で相互に属性やふるまいを利用しあう事で 1 つの機能を提供する．本研究では，ある部品がある部品を利用するとき，この部品間に利用関係が存在すると考え，部品グラフとしてモデル化する．部品を頂点，利用関係を有向辺で表したグラフを部品グラフと呼ぶ．以降，V を部品（頂点）の集合，E を有向辺の集合として，部品グラフを $G = (V, E)$ と表現する．コンポーネントランク [1] は利用頻度に基づいて部品グラフからそのソフトウェアにおける各部品の重要度となる評価値を算出する手法である．コンポーネントランクでは，部品グラフ $G = (V, E)$ 上の個々の辺および頂点に対して重みを繰り返し計算し，その後，対応する頂点の重みを各部品の評価値として，順位付けを行い評価を行う．重みとは，次のように定義される．

頂点の重み

部品グラフ G 上の各頂点 v は，$0 \leqq w(v) \leqq 1$ の重みを持つ．G の頂点の重みの総和は 1 とする．

辺の重み

頂点 v_i から v_j への辺 e_{ij} に関する辺の重み $w'(e_{ij})$ を式 (1) のように定義する．

$$w'(e_{ij}) = d_{ij} \times \ w(v_i) \tag{1}$$

d_{ij} は配分率と呼び，$0 \leqq d_{ij} \leqq 1$ かつ $\sum_i d_{ij}=1$ を満たす値とする．原点 v_i から v_j へ利用関係が存在しない場合，$d_{ij}=0$ とする．この配分率 d_{ij} は，頂点の重みの再計算において，有向辺の終点となる頂点の重みの決定に利用され，各頂点の重みが利用関係の先の頂点に分配される．

繰り返し計算の手順

1. 評価値の初期値として，各頂点に正の値を与える．
2. 値の変動が収束するまで，次の計算を繰り返す．収束後の対応する頂点の重みを各部品の評価値とする．
   - 辺の重みを現在の頂点の重みから決定する．
   - その頂点に入ってくる辺の重みの和を新しい重みとして，各頂点の重みを再計算する．

2.2　コードクローン関係を考慮したコンポーネントランクを拡張する方法

コードクローン [2] とは，ソースコード中での類似または一致した部分であり，同一の部品内だけにとどまらず，異なる部品間に存在する場合もある．コードクローンは無意味に出現するものでなく，同一処理や似た処理が必要になるなど，開発者の何らかの意図によって作りこまれる場合が多い．コードクローンはソフトウェアの保守工程においてプログラム管理の手間を増大させる可能性を持つので，コードクローンとなる部品を有する部品は，常に着目する必要がある．本研究においては，2 つ以上の部品のソースコードを比較したとき，30 トークン以上一致したコード片がある場合，その部品間にコードクローン関係があるとみなしている．千賀の研究では，コードクローンを持つ部品同士を部品グラフで結合する前後で，コンポーネントランクにおける各部品の評価値を比較すると，結合する部品から共通して利用される部品の評価値が下がることに着目した．さらに，結合する前後での各部品の評価値の変化を表示するツールを作成した [4]．これらの手法の有効性を確認するために，実際のオープンソースプロジェクトに対して適用を行い，評価値の変化が想定した通りに起こっているか [5]，評価値が減少した部品が，結合した部品において似たような方法で利用されているような部品であるか [6] を検証した．実験を行った結果，想定した事象を

確認できたことで，ある程度の有効性を確認したが，ツールなどで有効に利用するためには，さらなる精度の向上をはかることが必要であることを確認した．

3 パッケージ間距離を用いて結合を制限する手法によるコンポーネントランク拡張手法の改良について

3.1 実験の目的

[4]，[5] で行われた考察によると，関係の弱いコードクローンに基づいてノードの結合を行うと，関係の弱い部品同士の結合によって，部品グラフの構造が変化して，想定した評価値の変動が得られない場合が多く見られた．[7] では， パッケージ間距離が離れた部品間に存在するコードクローンの分析を行ったが，それらの中には関係の弱いコードクローンが多くみられる傾向があった．このことから，コードクローンを持つ部品間において，パッケージ間距離が一定以上離れた部品を結合しないことで精度を向上させることができると考えた．ここでいう精度の向上とは，似たような方法で利用されている部品が検出できるような状態のまま，その部品の評価値が下がっていることを指す．また，パッケージ間距離とは，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とする．実際に実験を行い手法の有効性を確認した上で，ツールにおける制限に利用する初期値として適切な値を提案する．

3.2 実験の手順

千賀の手法に基づいて，部品の結合前後で各部品の評価値が各結合条件でどのように変化するかを調査する．

1. 実際のオープンソースプロジェクトを収集し，各プロジェクトから 1 バージョン入手する．
2. CCFinder[3] を用い，コードクローン関係を抽出する．
3. Classycle[4] を用い，クラスの利用関係を抽出する．
4. 3 から部品グラフを作成し， 結合前のコンポーネントランクを計算する．
5. コードクローン情報を用いて， 部品グラフを結合する．クラス間のパッケージ間距離が 10，6，4，2，1 以上の場合は結合しないという条件をそれぞれ設定し，それぞれの条件で結合後の部品グラフを求める．部品グラフごとにコンポーネントランクを計算する．
6. それぞれのコンポーネントランクから各部品の評価値を求め，どう変化したかを分析する．

3.3 部品の分類について

コードクローン関係により結合されなかった部品に着目し，それらの部品をコードクローンにより結合した部品群からの利用関係を用いて，3 つのグループに分ける．以下の項目で評価する．

結合されなかった部品の分類

GroupA コードクローン関係により結合された部品群内の部品の 2 つ以上から共通して利用される部品 (以下 A)
GroupB コードクローン関係により結合された部品群内の部品の 1 つが共通して利用される部品 (以下 B)
GroupC コードクローン関係により結合された部品群から全く利用されていない部品 (以下 C)

3.4 評価項目

項目 1 A に属する部品数の変化

4 つのノード結合の条件それぞれを適用した結果の部品グラフに対し分類を行い，A に属する部品数の違いを調査する．想定される傾向として，パッケージ間距離の値が減少するとコードクローンの結合が起こりにくくなり，A の部品が減少することが考えられる．

項目 2 A，B，C に属する部品の平均変動率の変化

項目 1 と同様に 4 つのノード結合の条件で得られた部品グラフに対して分類を行い，A，B，C に属する部品の平均変動率を調査する．想定される結果として，不必要な結合が起こりにくくなることで A に属する部品の評価値がより減少することが期待できる．

変動率の定義

部品数により 1 つの頂点あたりの重みが異なるので，変動率にはグラフにおける頂点の総数を考慮する．ある頂点 v_i の頂点統合前のグラフでの評価値を $w(v_i')$，また，頂点統合のグラフにおける頂点の総数を $|V|$，頂点統合後のグラフにおける頂点の総数を $|V'|$ とする．このとき頂点 v_i の評価値の変動率を式 (2) で定義する．

$$\frac{w(v_i') \times |V'|}{w(v_i) \times |V|} \times 100 \tag{2}$$

項目 3 A に属する部品について，結合された部品群から同じように利用されているかどうか

5 つのノード結合の条件で得られた A の部品それぞれについて，結合された部品群から同じように利用されているかを調査する．図 1 はその例を示しており，V_1，V_2 は A に属している．V_1 は V_3，V_4 の結合している部品から同じように利用されているので該当するが，V_2 は該当しない．表では，A の部品中で該当する部品の数を原因部品数と表現する．また，適合率と再現率を定義し，各条件で A の部品のうち結合部品群から同じように利用されていた部品である割合と，各条件でそれらがどれくらい補足できていたかを示す．また，A の適合率と A の再現率の調和平均として F 値 [8] を求める．

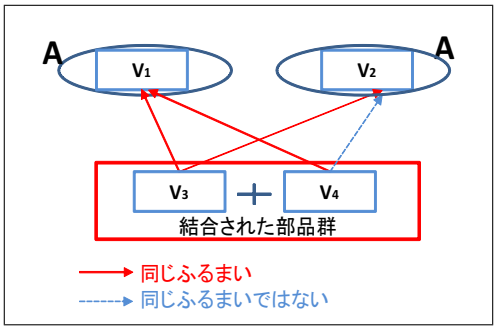
適合率と再現率の定義

A の部品数 N に対する，結合された部品群から同じように利用されている A の部品数 R の割合を適合率として式 (3) で定義する．

$$\text{適合率} = \frac{R}{N} \times 100 \tag{3}$$

同様に 5 つの条件のどれかで同じように利用されている A の部品の集合の要素 (要素数 C) に対して，各条件結合

図 1　結合部品群から同じように利用された部品



された部品群から同じように利用されている A の部品数 R の割合を再現率として式 (4) で定義する．

$$\text{再現率} = \frac{R}{C} \times 100 \tag{4}$$

4　結果

4.1　全体的な傾向

調査対象として，31 個のオープンソースプロジェクトを入手した．これらの中には，次のような事例が存在し，これら 23 プロジェクトでは，パッケージ間距離の制限を設定しても結果が変わらなかった．

1. 全てのファイルが同一パッケージ内に存在しているプロジェクト (6 プロジェクト)
2. パッケージは複数存在するが，コードクローンが全く存在しなかったプロジェクト (2 プロジェクト)
3. パッケージは複数存在するが，同じパッケージ内でのコードクローンしか存在しなかったプロジェクト (6 プロジェクト)
4. パッケージは複数存在し，異なるパッケージにまたがるコードクローンも存在したが，その部分からは A の部品が生成されなかったプロジェクト (9 プロジェクト)

一方，実際に変化がみられた 8 プロジェクトにおいては，パッケージ間距離の制限を強くするにつれて A の部品数や，同じように利用されている A の部品数が減少していった．以降では，変化があった 8 プロジェクトの中から 3 つの事例を中心に紹介する．

4.2　事例の紹介

Card Me は電子名刺の標準規格フォーマットである VCard ファイル形式の電子名刺を読み書きすることができるソフトウェアであり，JavaSIP は，IP 電話のサービスで標準となっているプロトコルを実装したソフトウェア，JID は，画像ホスティングサービス用の画像ダウンローダである．表 1 は Card Me を対象とした場合の結果，表 2 は JavaSIP を対象とした場合の結果，表 3 は JID を対象とした場合の結果をそれぞれ示している．

表では，パッケージ間の各距離に対する結果ごとに，A の部品数，コードクローンにより結合された部品数，A，B，C に属する部品の平均変動率， A の原因部品数と適合率，再現率，F 値を紹介している．これらの表から，以下のような結果が得られた．

- パッケージ間距離の制限を強くするごとに，A に属する部品の平均変動率も減少している．このことから，不必要な結合が行われなくなり，想定したような結果が得やすくなっていると考えられる． Card Me や JavaSIP では，パッケージ間距離の制限を強くするごとに段階的に結合している部品が減少し，それにつれ

表 1　Card Me に対する分析結果

パッケージ間距離	10	9〜4	3	2	1
A の部品数	33	31	32	12	9
結合部品数	55	54	53	50	46
A の平均変動率	99.2 %	98.2 %	96.4 %	59.8 %	38.9 %
B の平均変動率	112.7 %	113.3 %	113.7 %	114.5 %	111.8 %
C の平均変動率	121.1 %	120.0 %	120.5 %	112.7 %	111.9 %
原因部品数	23	22	22	9	7
適合率	70.0 %	70.0 %	69.0 %	75.0 %	78.0 %
再現率	100 %	96.0 %	96.0 %	39.0 %	31.0 %
F 値	82.4 %	81.0 %	80.3 %	51.3 %	43.3 %

表 2　JavaSIP に対する分析結果

パッケージ間距離	10〜6	5〜3	2	1
A の部品数	26	20	20	18
結合部品数	17	16	13	4
A の平均変動率	89.1 %	88.2 %	88.9 %	87.9 %
B の平均変動率	102.1 %	102.3 %	103.7 %	103.5 %
C の平均変動率	94.7 %	95.1 %	96.3 %	96.0 %
原因部品数	7	5	5	3
適合率	27.0 %	25.0 %	25.0 %	17.0 %
再現率	100 %	71 %	71 %	43 %
F 値	42.5 %	37.0 %	37.0 %	23.9 %

表 3　JID に対する分析結果

パッケージ間距離	10〜5	4〜1
A の部品数	6	2
結合部品数	4	2
A の平均変動率	93.9 %	94.2 %
B の平均変動率	96.6 %	96.1 %
C の平均変動率	100.7 %	100.5 %
原因部品数	4	1
適合率	67.0 %	50.0 %
再現率	100 %	25.0 %
F 値	80.2 %	33.3 %

て A の部品数が減少している．一方，JID では制限を強めたある段階でのみ変化した．

- パッケージ間距離の制限を強くするごとに，結合された部品群から同じように利用されている A の部品数も減少している．特に，1，2 以上の条件下では，大きく減少している．パッケージ間距離が離れた部品においても，コードクローン中で共通して利用されている部品が存在し，パッケージ間距離が 1，2 より離れたクラス間にもコードクローンも相当量存在している．
- パッケージ間距離の制限を強くすることで適合率が向上することで想定していたが，想定に反し，あまり適合率は変化しなかった．一方，再現率については想定通り，減少し，ある制限より強くすると再現率が大きく減少した．これらのことから，F 値はパッケージ間距離の制限の強化に従って，減少した．

5　考察

31 プロジェクトに対する結果，23 プロジェクトでは十分な効果が得られなかった．このことから，本手法が効果を発揮するためには，部品のパッケージ分けが行われているようなある程度の規模が必要である．

残った 8 プロジェクトでは，パッケージ間距離の制限を強くしていくことで，結合する部品数の減少に伴い，A の部品数も減少した．また，A の部品の平均変動率は制限を強くすることで下がる傾向にあり，不必要な結合が起こりにくくなることで，想定した結果がある程度得られた．再現率を考慮せず，パッケージ間距離が小さい部品のみを対象とする場合，本手法を用いて制限を強くすることは対象となる部品を評価値の変動率を用いて抽出するのに有効であると考える．

一方で，本手法を用いた場合，A が結合された部品群から共通して利用されている部品である割合は，制限を強くしてもあまり変動しなかった．これは，パッケージ間距離が離れた部品間に共通して利用されている例が，少なからず存在することを示している．F 値による観点からは，制限を加えないほうが結果的に良いという結果を得られた．

これらの結果より，平均変動率と F 値の間には，トレードオフの関係が存在していると考えられ，手法を考えると F 値が大きく減少しない範囲内で，平均変動率が大きく減少している条件を選ぶのが最善であると考えられる．実際に 8 プロジェクトをそれぞれ考慮すると

- パッケージ間距離の制限が 3 以上の場合は結合しないという条件が最善なプロジェクト（6 プロジェクト）
- パッケージ間距離の制限が 5 以上の場合は結合しないという条件が最善なプロジェクト（2 プロジェクト）

と分類できたので，それらの値を初期のしきい値候補として，設定するのが良いと考えられる．これらの結果は，[7] において確認できた，共通した利用方法をもつコードクローンをもつクラスの最大のパッケージ間距離が，多くのプロジェクトで 2 または 4 であったことと整合していると考えられる．

6　まとめ

本研究では，結合対象となる部品同士がパッケージ配置上で，どれだけ離れているかの情報に基づいて，一定以上離れた部品を結合しないことで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精度が向上するかを検証した．結合の条件を厳しくすることで，想定した結果だけが残りやすくなり，対象となる部品の評価値が下がりやすくなることを確認した．しかし同時に，結合部品が減少することで，結合した部品から共通して利用される部品の一部が検出できなくなることも確認した．一般的な利用においては，それらのバランスを考慮したしきい値の設定が必要である．今回の分析結果に基づいて，分析のしきい値を設定することで，手法の精度が向上すると考えられる．今後の課題として，さらなる精度の向上を目的として，他の改良手法の適用があげられる．

参考文献

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

[2] 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.

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

[4] 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，2014.

[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，2015.

[6] 安藤 正憲，堀江 大河，井田 裕之 : ” 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 ― 変化の影響を受けた部品について分析 ―”，南山大学情報理工学部 2014 年度卒業論文，2015.

[7] 野々山 未菜 : ” 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”，南山大学情報理工学部 2014 年度卒業論文，2015.

[8] 徳永 健伸 : ” 情報検索と言語処理”，東京大学出版会，1999.