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～

2009SE229　奥田黎哉　2011SE132 小林登夢　2011SE179 村瀬慶紀
指導教員：横森励士

1　はじめに

近年のソフトウェアは規模が大規模化しており，それに伴いソフトウェア部品の数も増大し，内部の関係も複雑化している．このような大量の部品下においては，ソフトウェアをモデル化し，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う方法を用いることで，目的に沿った部品を効率的に選び出せると考えられる．千賀は部品間の関係をモデル化し，部品の評価をおこなう手法であるコンポーネントランクにコードクローンの関係を取り込むことで拡張する手法 [1] を提案した．この方法は既存のコードクローンが共通して利用している部品を抽出するのに有効であると思われるが，現段階では小規模の仮想ソフトウェアでのみしか検証を行っておらず，一般的なソフトウェア部品グラフにおいても有効であるかどうかを検証していない．

本研究では，一般的なソフトウェアに対する提案手法の有効性を確認するために評価実験を行う．実際に存在する多数のオープンソースプロジェクトのソースコードに対して提案手法を適用し，適用前後で部品グラフの評価値がどう変化するかに着目する．コードクローンの関係を取り込む前と後での評価値の変動率の変化の割合や，変動率の平均の算出という評価実験を行い，結果の妥当性を考察する．この結果を元に，提案手法が一般的なソフトウェアに対しても有効であることを確認する．

2　背景技術

2.1　部品グラフ

一般に，ソフトウェア部品とはモジュールや関数，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振る舞いを利用しあうことで一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時，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．

2.2　コンポーネントランク

コンポーネントランク [2] 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計算し，対応する頂点の重みを各部品の評価値として，順位に基づいて評価を行う．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 $0 \leqq w(v) \leqq 1$ の重みを持ち, 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 $w'(e_{ij})$ 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 $\Sigma_i d_{ij} = 1$ を満たす値とする．頂点 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，次に示す頂点の重みの定義において，有向辺の終点となる頂点の重みの決定に利用される．

頂点の重みの再計算

$IN(v_i)$ を v_i を終点とする有向辺の集合とする．この時，頂点 v_i の重みは v_i が終点となる有向辺 e_{ki} の重みの総和とする． 式 (2) で頂点の重みを再計算する．

$$w'(e_{ij}) = \sum_{e_{ki} \in IN(v_i)} d_{ij} \times \ w(v_k) \tag{2}$$

繰り返しの計算の手順

1. 評価値の初期値として，各頂点に正の値を与える．
2. 値の変化が一定以下になるまで，次の計算を繰り返す．
   (a) 辺の重みを現在の頂点の重みから決定する．
   (b) 頂点の重みを再計算する．
3. 収束後の各頂点の重みを対応する部品の評価値とする．

2.3　コードクローン関係をコンポーネントランクに反映させる手法について

コードクローンとは，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はなく，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．

先行研究 [1] では，コンポーネントランクの拡張方法としてコードクローンの関係を持つ頂点の統合を行う手法を提案した．小規模な仮想ソフトウェアを対象としてコードク

ローンを考慮した評価値計算を行い，評価値の変化を調査した．さらに，実際のソフトウェアにおいて評価値が減少する部品がコードクローンを持つ部品から共通の方法で利用される部品であるかどうかを調査した．これにより以下の結果が得られた．

- 小規模な仮想ソフトウェアを対象とした実験では，部品グラフ上で部品の統合が行われた時，統合された部品の 1 つからのみ利用している部品や関係のない部品よりも統合された部品群の 2 つ以上の部品から共通して利用されている部品の方が評価値が大きく減少する．
- 評価値が減少する部品がコードクローンを持つ部品から利用される部品となっている事例が存在する．

ただし，この研究では評価値の減少について小規模の仮想ソフトウェアでのみしか検証を行っておらず，分析としては不十分である．先行研究 [1] の手法が一般的に有効であることを示すために，より多くのプロジェクトについて調査する必要がある．

3 コンポーネントランク拡張手法の有効性評価

3.1 目的

本研究では，先行研究 [1] の手法の有効性を評価するために，部品グラフ上で部品の統合が行われた時，1 つからのみ利用している部品やその他の部品よりも 2 つ以上の部品から共通して利用されている部品の方が評価値が大きく減少することが実際のソフトウェアにおいても成り立つのかを調査する．

3.2 分析手順

(1) オープンソースのプロジェクトを収集し，各プロジェクトからソースコードを 1 バージョン分入手する．

(2) CCfinder[3] を用いてクローン関係を，Classycle[4] を用いて利用関係を解析する．

(3) (2) で得たクローン関係，利用関係を元に，千賀が提案した手法 [1] を適用し，部品一覧と結合前後の各部品のコンポーネントランクを求める．

(4) 適用前後の部品グラフから，結合した部品群それぞれに着目し，結合されなかった部品を次のように分類する．

GroupA 統合した部品群のどれかから利用されており，ある部品群内の 2 つ以上の部品から利用されている部品．

GroupB 統合した部品群のどれかから利用されているが，どの部品群からも高々 1 つの部品からしか利用されていない部品．

GroupC 統合した部品群から全く利用されていない部品．

(5) 部品グラフの評価値の変化として以下の項目を分析する．

1. プロジェクト単位で見た時の A，B，C の各グループの部品の評価値の変動率の平均を調査する．プロジェクト毎に部品数が異なるので，プロジェクト単位の平均変動率という観点から分析する．
2. プロジェクト単位で見た時の GroupA と GroupB に属する部品の評価値の平均変動率の差，GroupA と GroupC に属する部品の平均変動率の差を調査する．これにより GroupA に属する部品が GroupB，C に属する部品よりも評価値の減少が大きいことを確認する．
3. プロジェクト毎に評価値の減少が大きくなる部品の Top5，Top10，TOP15，TOP20 を選出し，その中での GroupA，GroupB，GroupC に属する部品の割合を調査する．これにより評価値が大きく下がる部品の傾向を確認する．

3.3 評価値の変動率

評価値の増減の判定には，統合前後の各頂点の重みの変動率を用いる．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 $w(v_i)$，頂点統合後のグラフでの評価値を $w(v_i')$，また，頂点統合前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（3）で定義する．

$$\frac{w(v_i') \times |V'|}{w(v_i) \times |V|} \times 100 \tag{3}$$

4 評価値の変動率の調査

リストアップした 45 個のプロジェクトに対し，分析手順を元に調査を行った．実験ではソースコード中の各クラスを部品として扱っている．

4.1 プロジェクト単位での評価値の変動率

プロジェクト毎に部品数が異なることを考慮し，ABC 各グループに属する部品の平均変動率を求めた．それらの平均・標準偏差を表 1，各グループの変動率の分布を図 1，図 2，図 3 に示す．GroupA，B，C が存在しなかったプロジェクトがあるのでプロジェクト数は 45 より小さい数となっている．

表 1 より，手法の適用前後での変動率の平均は GroupA の減少が最も大きく，GroupB と GruopC はあまり変化は見られなかった．また変動率が減少したプロジェクトの割合は，GroupA が約 93% と最も高く，GroupB は約 54%，と GroupC は約 53% となった．

図 1, 2, 3 より，GroupA，B，C の各プロジェクトの平均変動率の分布を比較すると次のようなことが確認出来る．

- GroupA の分布の山は，GroupB，C よりも左に存在する．ただし，GroupA に属している部品でも評価値が大きく増加した部品が一部あり，平均値を押し上げる結果となったプロジェクトも存在する．

- GroupB の分布をみると，大部分のプロジェクトが 90% 以上となっている．ただし大きく減少したプロジェクトも存在する．
- GroupC の大部分のプロジェクトは，90% から 110% の範囲に存在している．

表 1　各プロジェクトの評価値の変動率の平均と標準偏差

	プロジェクト数	平均	標準偏差	評価値の増減	
				増加	減少
GroupA	43	88.5%	19.7	3	40
GroupB	41	107.8%	16	19	22
GroupC	43	104.3%	6.4	20	23

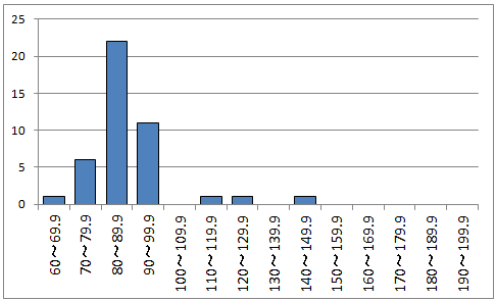

図 1　プロジェクト単位での GroupA の分布

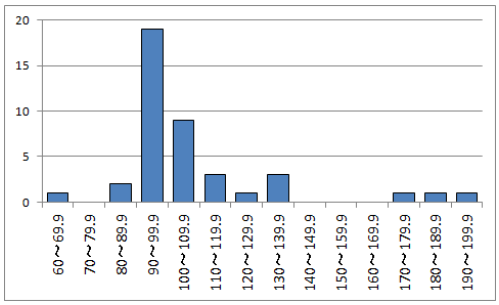

図 2　プロジェクト単位での GroupB の分布

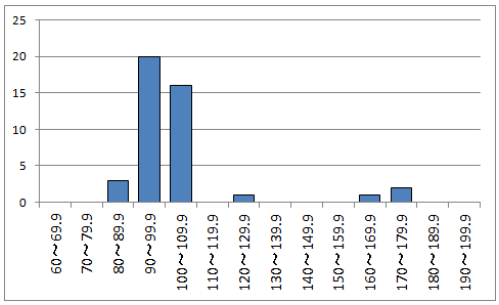

図 3　プロジェクト単位での GroupC の分布

4.2　プロジェクト単位で見た時の変動率の差

GroupA と GroupB が両方とも存在する 36 のプロジェクトに対して A の平均変動率と B の平均変動率の差，GroupA と GroupC が両方とも存在する 37 のプロジェクトに対して A の平均変動率と C の平均変動率の差をそれぞれ求めた．結果を表 2 に，B-A の変動率の分布，C-A の変動率の分布をそれぞれ図 4，図 5 に示す．

表 2 より，B-A の平均変動率の差の平均と C-A の平均変動率の差の平均はともに 15～20% のプラス値になり，GroupA に属する部品の評価値が下がりやすいことが分かる．この結果が統計的に有意な差があるかを Welch の t 検定を用いて，GroupA と GroupB，GroupA と GroupC の二つの対応するグループにおける評価値の変動率を比較したところ，GroupA と GroupB についての t 検定 (α =0.01) では，効果量 d=1.04000，検定力 1-β =0.9998752，一方，GroupA と GroupC についての t 検定 (α =0.01) では，効果量 d=0.94610，検定力 1-β =0.9993245，といずれの場合も十分大きな値が得られ，有意な差があることを確認できた。これらの結果から，統合した部品群のどれかから利用されており，ある部品群内の 2 つ以上の部品から利用されている部品は 1 つのみ，あるいはその他の部品に比べ評価値が大きく減少することが成り立つことを示している．一方で，GroupA の評価値の低下より，B，C の評価値の低下の方が大きいプロジェクトも 1 割強存在したので，それらの原因も調査し考察する．

表 2　プロジェクト単位で見た時の変動率の差

	プロジェクト数	差の平均	標準偏差	0 以上	0 未満
B-A	36	20.8%	20	3	33
C-A	37	15.8%	16.7	5	32

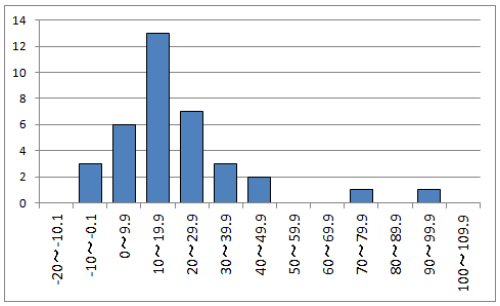

図 4　B-A の変動率の分布

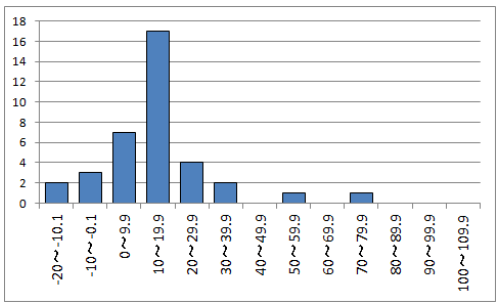

図 5　C-A の変動率の分布

4.3 プロジェクト毎の TOP5，TOP10，TOP15，TOP20

各プロジェクトで，評価値が減少した部品のみを対象に，その中で減少度合いの大きい部品の TOP5 と TOP10，TOP15，TOP20 までを抽出し，それらの部品が GroupA，B，C のどれに属するかを調査した．ただし GroupA の部品が全て抽出された段階で，それ以下の順位の部品は無視している．結果を表 3 に示す．表 3 より，TOP5 の部品が属する割合では，GroupA が 76.8% と大部分を占めるが，TOP10，TOP15，TOP20 を対象にした場合，その割合が下がることも確認した．

表 3 TOP5，TOP10 の部品が属する GroupA，GroupB，GroupC の割合

	総部品数	GroupA	GroupB	GroupC
TOP5	164	76.8%	17.7%	5.5%
TOP10	267	73.4%	19.9%	6.7%
TOP15	391	62.4%	20.5%	17.1%
TOP20	501	53.7%	20.6%	25.7%

5 考察

5.1 プロジェクト単位での評価値の変動率

図 1 より，グループごとの評価値の平均値を押し上げる結果となっているプロジェクトが 3 つ存在することが分かり，プロジェクト単位で原因を調査した結果，以下のことがわかった．

- 統合した部品の評価値が統合する前と比べて大幅に増加したことで利用先の評価値も大きく増加した．
- 統合した部品内で他の部品を大量に使っている部品が存在し，その部品は統合した内部の部品も使っていたことから，統合した内部の部品が他の内部の部品を大量に使用していたことにより，利用先の評価値が大きく増加した．

以上のことから，部品グラフの内部構造を変化が大きい場合，想定以上に評価値が変動する場合があるということがわかった．

5.2 プロジェクト単位で見たときの変動率の差

GroupA の評価値の低下より，B，C の評価値の低下の方が大きいプロジェクトが 3 つ存在した．この 3 つのプロジェクトの共通点を調査すると，統合する部品間の距離が遠く，部品グラフの内部構造が劇的に変化しすぎたためと推測できた．そこで，3 つのプロジェクトに対し，パッケージ間の距離が 2 以内の部品を統合するよう再設定し分析した．ここで距離とはあるパッケージ階層からあるパッケージ階層まで到達するためのパッケージ階層の移動回数である．その結果を次の表 4，5，6 に示す．

表 4，5，6 より A の評価値は減少し B，C の評価値は増加の傾向にあることがわかる．よってこの手法を用いる場合，プロジェクトによっては距離の設定を適切にすることで精度が向上する可能性があることがわかる．

表 4 CardMe

	部品数	変化前変動率	A との差	変化後部品数	変化後変動率	A との差
a	14	94%	0%	3	70.4%	0%
b	3	93.1%	-0.9%	10	113.9%	43.5%
c	19	94.7%	0.7%	24	92.3%	22%

表 5 jackcess

	部品数	変化前変動率	A との差	変化後部品数	変化後変動率	A との差
a	23	96.9%	0%	21	94.3%	0%
b	32	91.2%	-5.7%	37	98.7%	4.4%
c	55	84.7%	-12.14%	56	99.6%	5.3%

表 6 vietime

	部品数	変化前変動率	A との差	変化後部品数	変化後変動率	A との差
a	10	93.1%	0%	4	69.6%	0%
b	5	90.2%	-2.9%	12	92%	22.4%
c	6	93.1%	-0.1%	41	99.3%	29.7%

6 おわりに

本研究では，実際のオープンソースプロジェクトのソースコードを対象に，コンポーネントランク計算において，部品グラフ上のコードクローン関係を持つ部品を統合する手法を適用し，適用前後で統合する内部の 2 つ以上の部品から利用されている部品は，統合する内部の部品の 1 つからのみ利用している部品や関係のない部品より，大きな評価値の減少が見られることを確認した．結果からは，既存のコードクローンが共通して利用している部品を検出する仕組みを実現する上で，部品グラフ上の評価値の変化という観点からは，コンポーネントランク計算において，部品グラフ上のコードクローン関係を持つ部品を統合する手法が有効であると考えられる．

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，2014.

[2] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, and ShinjiKusumoto.”Ranking Significance of SoftwareComponents Based on Use Relations”, IEEE Trans. SoftwareEngineering, vol. 31，no. 3，pp. 213-225，2005.

[3] Toshihiro Kamiya，Shinji Kusumoto，and Katsuro Inoue.”CCFinder:. A Multi-Linguistic Token-based Code Clone Detection System for Large ScaleSource Code”, IEEE Trans. SoftwareEngineering, vol. 28, no. 7，pp. 654-670，2002.

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

[5] ” 検定力分析のすすめ,”
http://www.relak.net/psy/power/.