利用先や利用元の部品の共通性に基づく ソフトウェア部品分類手法の提案

2012SE047 堀貴行　2013SE022 後藤彗
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ており，ソフトウェアを構成する部品の数も増大している．このような環境下では，部品間の類似性などを利用してソフトウェアの構成要素を効率よく把握することが求められる． 橋倉らは利用関係やコードクローン関係に関するメトリクス値が似ている部品を抽出する手法を提案した [1] が，分類対象はある程度値の大きいものとなってしまう．より精密に似ている部品を分類し抽出するには，共通性が明確に表れているような情報に基づいて分析を行う必要がある．

本研究では，各部品ごとに利用先部品の集合と利用元部品の集合を求め，部品対ごとに部品間の類似度をそれぞれ評価する．各部品間の類似度をもとに階層的クラスター分析により分類を行い，同じような機能を実現する部品，同じところから利用されている部品を抽出する手法を提案する．階層的クラスター分析後に得られた分類結果を調査し，提案手法による分類の有効性を評価する．提案手法を用いてソフトウェア部品の分類を行うことで，ソフトウェアの部品構成を理解する際にひとまとめに部品群を抽出することができると考えられる．

2　関連研究

2.1　ソフトウェア部品グラフと利用関係

ソフトウェア部品とは，内容をカプセル化したうえで，ソフトウェアを実現する環境において交換可能な形で配置できるようにしたシステムモジュールの一部をさす [2]．本研究では，Java ソフトウェアを対象とし，それぞれのクラスのソースコードが記述されるファイルを部品の単位として部品グラフをモデル化する．部品グラフ上の頂点は各部品を表し，辺は部品間の関係を表現する． 部品グラフ上で表現する関係として部品間の利用関係を考える．Classycle[3] を利用しクラスの継承，インスタンスの生成，メソッドの呼び出し，フィールド参照を利用関係として抽出し，部品グラフ上で利用元の部品から利用先の部品への有向辺で表現する．

2.2　メトリクスを用いた類似したソフトウェア部品の抽出手法について

ソフトウェアを構成する部品の特徴を表すメトリクスを抽出して類似したメトリクスを持つ部品を抽出する方法が提案されている．小堀ら [4] はトークン数の出現回数などを比較し，それらの多くが一致した部品同士を類似部品とみなす方法を提案した．また Wise ら [5] は学生が提出した課題の盗用を検出するシステムを提案し，キーワードで構成されたトークン文字列を比較することでコピーを検出した．これらの研究では比較対象を減らすために総トークン数を主メトリクスとして用いて主メトリクスが似ている部品のみと比較することで，大量のソフトウェアから得られた部品群の中から，コピーなどによって生成された部品を検出するための方法として有効であることを確認している．一方で橋倉ら [1] は，一つのソフトウェアで構成する部品の集合を対象としてそれぞれの部品の利用関係やコードクローン関係のメトリクスを抽出し，それらの 2 つを組み合わせて値の似ている部品群を抽出することで，他の部品の利用の仕方が似ている部品や，同じクローンセットに含まれるような部品をさらに別の観点から分類することを提案した．提案された分類方法により，同じ特徴を示した部品ごとに共通した関係のある部品を調べることで，分類した結果に意味付けを行うことができ，意味がある分類結果としてひとまとめに抽出できることを示した．

3　利用先や利用元の部品の共通性に基づくソフトウェア部品の分類手法の提案

3.1　研究の動機

過去の研究 [1] では， 利用関係やコードクローン関係に関するメトリクス値が似ている部品を抽出することで，役割が似ている部品を抽出する手法を提案している． この手法の場合，数だけで評価を行うので偶然個数が一致したものまで含めてしまう．また，値が小さい部品は関係がなくても一緒にまとまることが多く，分類対象はある程度値の大きい部品に限られてしまう．私たちは，ただ単に数だけでなく，利用先や利用元部品で一致しているという情報はより強い関連性を示していると考えた．利用先の部品に共通性があるということは，同じ機能を利用していたり，同じ対象を扱っているなどの共通性を有していると考えられる．一方で，利用元の部品に共通性があるということは，セットで利用されることが多い部品と認識できる可能性が高いと考えられる．これらの情報をもとに分類を行うことで，強い関連性をもった部品集合に分けることができると考えた．

3.2　提案手法のアプローチと実現した環境

提案手法に基づく分類を行うために，ある 1 つのソフトウェアを構成する部品群を分析対象として，次のような手順で利用関係の抽出と部品間の類似度の計算を行う．実現したツールの構造を図 1 に示す．ツールでは部品間の類似度から生成した行列を出力する． 出力された行列を距離

行列として階層的クラスター分析を用いて分類を行う．出力された樹形図から部品群を読み取り，それぞれのグループの特徴を確認し，1 つのソフトウェアを構成する部品がどのように分類されたかを確認する．

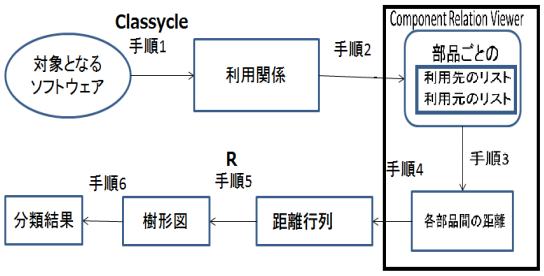

図 1　ツールの構造

1. 分析対象のソフトウェアを Classycle[3] で分析して，クラス間の利用関係を入手する．
2. 1 で入手した利用関係から，各部品の利用元部品の集合，利用先部品の集合を作成する．
3. 利用元部品もしくは利用先部品の類似度のどちらで分類するかを決め，部品対ごとに類似度と距離を計算する．
4. 各部品間の距離を示す距離行列を作成する．
5. 距離行列を用いて階層的クラスター分析を行い樹形図を得る．
6. 得られた樹形図において，まとまりになっている部分から類似部品群を抽出する．

3.3　類似度と距離行列の生成について

ある部品 A の利用元 (もしくは利用先) 部品の集合を L_A，ある部品 B の利用元 (もしくは利用先) 部品の集合を L_B としたとき，それらの集合間の類似度 $sim(L_A, L_B)$ を Jaccard 係数を用いて示す．

$$sim(L_A, L_B) = \frac{|L_A \cap L_B|}{|L_A \cup L_B|} \tag{1}$$

この値は 0〜1 の値域を持ち，高いほど類似していることを示しているので，距離 $dist(L_A, L_B)$ を以下のように定義し，距離行列の作成に用いる．

$$dist(L_A, L_B) = 1 - sim(L_A, L_B) \tag{2}$$

4　評価実験

6 つのオープンソースソフトウェアを対象に提案手法を適用した．対象のソフトウェアに対して利用先部品の類似度と利用元部品の類似度でそれぞれクラスター分析を行い，部品群を得る．得られた部品群内の部品がどのような関係を持っているかを後述の基準に基づいて確認する．利用先部品の分類と利用元部品の分類，それぞれの手法で得られた部品の集合がどのような特徴を持っているかを調査することで，分類手法の特徴を確認する．

4.1　類似度の定義について

クラスター分析を用いて得られた部品群内の部品がどのような関連性をもっているかを確認するために 6 つの基準を設定し判定する．

基準 1　部品の派生元が同じであったり，実装しているインターフェースが同じである．派生元が同じであることや，共通の機能をもっているという観点から類似していると判断する．

基準 2　分類された部品群が同じパッケージに所属している．開発者が同種の目的と判断し，パッケージ構造のなかで部品の整理を行った結果と一致していると考える．

基準 3　分類された部品の扱う対象が同じである．必要な前処理や後処理が一致しているなど同種の処理が行われている部品として関連していると判断する．例えば，あるオブジェクトの視点を移動させる部品とあるオブジェクトの視点を回転させる部品のように，操作の対象となるオブジェクトが同一もしくは同じ種類である場合のとき，そのオブジェクトを操作するための部品として類似しているとみなす．

基準 4　分類された部品の役割が同じである．分類された部品群が 1 つの特定の役割を果たす部品でまとまり，同じ目的の処理が行われている部品として関連していると判断する．例として，ステージを照らす部品と物体を照らす部品が存在し，異なるオブジェクトに対して同一の操作を行う場合のとき，扱う対象が異なる場合でもその部品が実現する機能や役割が同じであるとみなす．

基準 5　部品の役割や対象やパッケージなどから 1 つの機能群としてまとめられると考えられる部品群である．1 つの大きな役割に対してその一部を実現する部品群であるということで関連していると判断する．例として，keyframe を管理する際に利用する部品群のように役割や対象が異なる場合でも全体として keyframe を管理するという 1 つの機能を実現するため部品群として認識できる場合，1 つの機能群を構成する部品とみなす．

基準 6　共通の利用元を考慮することで，その利用元からの機能群として認識できる．基準 5 と比較して，利用元の情報が認識のために必要であるという点が異なり，主に利用元部品の分類を考えるときに利用する．例として，物体の関節部を編集する部品と物体の回転を編集する部品だけでは機能群としてみなすのは難しいが，3D のモデルをメッシュで表現し編集する機能を持つ MeshEditor という共通の利用元部品を加えた場合，メッシュの制御を行う機能群として認識できる

利用先部品の分類									
	全体部品数	クラスター数	分類した部品数	意味付けできた部品数	基準1	基準2	基準3	基準4	基準5
Art Of Illusion[6]	190	13	56	56	8	11	4	8	9
Turtle Sport[7]	386	36	120	120	32	30	28	30	34
GeoAPI[8]	257	27	141	141	23	16	18	21	23
PixEditor[9]	103	14	40	40	14	11	8	9	14
jlGui[10]	70	4	16	16	3	4	3	4	4
Card Me[11]	118	5	41	41	5	3	3	5	5

利用元部品の分類										
	全体部品数	クラスター数	部品数	意味づけできた部品数	基準1	基準2	基準3	基準4	基準5	基準6
Art Of Illusion[6]	190	17	71	71	7	9	7	5	8	15
Turtle Sport[7]	386	46	131	131	13	19	23	11	14	40
GeoAPI[8]	257	34	86	86	23	26	11	5	17	31
PixEditor[9]	103	11	44	44	4	5	3	2	8	10
jlGui[10]	70	12	29	29	5	9	1	2	7	12
Card Me[11]	118	8	54	54	3	5	2	4	2	8

表 1 オープンソースソフトウェアの分類結果

	基準1	基準2	基準3	基準4	基準5
グループ1	○	×	○	○	○
グループ2	○	○	○	○	○
グループ3	○	×	○	○	○
グループ4	○	○	×	○	○
グループ5	○	○	×	○	○

表 2 利用先部品の意味付けの結果

ので，この基準を加えた．

4.2 利用先部品に基づく分類の結果

表 1 は実験の結果をまとめたもので，1 つのソフトウェアを構成する部品の総数，提案手法によって分類対象となったクラスターの総数，分類対象となったクラスター内にある部品の総数，基準 1 から基準 6 のいずれかに当てはまったクラスター内の総部品数と各基準にあてはまった部品数を示している．また基準を満たしているクラスターの割合によって色分けしている．最も濃い色のセルは 7.5 割以上，薄い色のセルは 5 割以上 7.5 割未満，白色のセルは 5 割未満のクラスターが基準を満たしたことを示す．

表 1 上部の利用先部品に基づく分類では，全体の 3 割前後の部品が分類対象となり，分類対象となった部品は基準 1〜5 のいずれかには当てはまることがわかった．一つのクラスターには平均すると 3〜8 個の部品で構成されており，多くのクラスターの部品は基準 1 と 5 を満たしていた．同じパッケージ内に存在している部品が多く機能群としてまとまった分類がすでになされていることがわかる．機能群としてみることができるが，扱う対象が異なる場合も多く，基準 3 を満たさないケースが多くみられた．

個別のプロジェクトに対する分類結果として，図 2 で Card Me[11] 内の部品を分類した結果を示す．横線より下の部分は類似度が 0.5 以上である組み合わせとなっており，5 つの部品群を抽出できた．抽出できたそれぞれの部品群に対して，どの基準を満たしたかを表 2 で判断した．

グループ 1 はパッケージは異なるが VCard という部品を対象としてクラスの継承元の補助を行う機能群として見

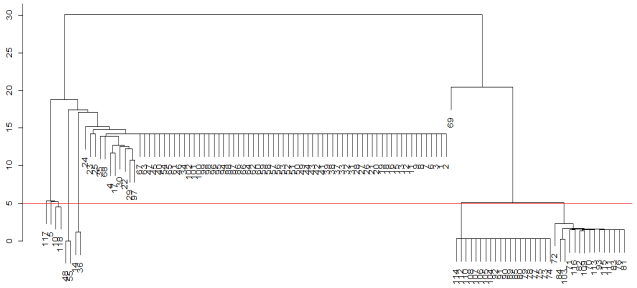

図 2 利用先部品の類似度で作成した樹形図

ることができた．グループ 2 はすべての部品がインターフェースであり，メディアファイルを対象としている．また利用先部品がすべて一致していて，継承しているクラスも同じであるなど機能群としてまとまっていることがわかる．グループ 3 は部品間の距離がある程度はなれているが片方の部品が利用先としてもう一方を含んでおり，ISOUtil を対象にフォーマットの設定をする役割で共通しているため，関係があると推測した．グループ 4 と 5 は長さの格納を行う機能を持ち，扱う対象は様々であるがクラスの継承が一致していて，利用先部品が多く一致しているため，機能群としてのまとまりがあることがわかる．このように，利用先部品が多く一致している集合は，部品同士の役割が同じであったり，機能群としてまとまりを確認でき，関連性を強く認識できる部品の集合となる．

4.3 利用元部品に基づく分類の結果

利用元部品に基づく分類の結果を表 1 下部に示す．利用先による分類の場合と同じく，全体の 3 割前後の部品が分類対象となり，分類対象となった部品は主に基準 6 に当てはまっていることがわかった．一つのクラスターには平均すると 2〜7 個の部品で構成されており，利用先部品に基づく分類と比べると，クラスター数，分類できた部品数共に多くなっている．

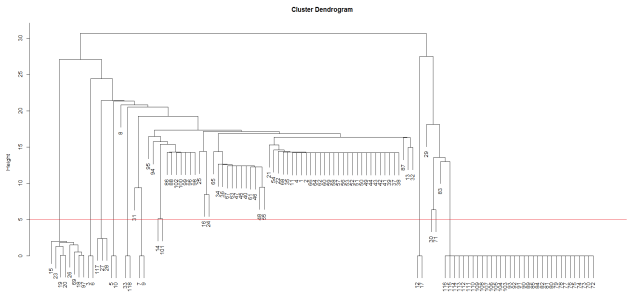

図 3　利用元部品の類似度で作成した樹形図

	基準1	基準2	基準3	基準4	基準5	基準6
グループ1	×	○	○	○	×	○
グループ2	×	×	×	×	×	○
グループ3	×	○	×	○	○	○
グループ4	×	×	×	○	○	○
グループ5	○	○	×	×	×	○
グループ6	○	○	×	×	×	○
グループ7	○	×	×	×	○	○
グループ8	×	○	○	○	×	○

表 3　利用元部品の意味付けの結果

個別のプロジェクトに対する分類結果として，図 3 で Card Me[11] 内の部品を分類した結果を示す．横線より下の部分は類似度が 0.5 以上である組み合わせとなっており，8 つの部品群を抽出できた．抽出できたそれぞれの部品群に対して，どの基準を満たしたかを表 3 で判断した．

グループ 1 と 8 は Type クラスの部品でまとまっていて，パッケージと部品の役割が同じであった．グループ 2 は VCard を含めるとユーザーインターフェースを実装する機能群としてみることができた．グループ 3 と 4 は画面に機能を表示をするための部品がまとまっていて機能群としてみることができた．グループ 5 と 6 は Scheme クラスの部品でまとまっていて，値を受け取ってから必要な情報を返す役割を持っていた．グループ 7 はパッケージは異なるが VCard という部品を対象としてクラスの継承元の補助を行う機能群として見ることができた．また，すべてのグループは共通の利用元部品に VCard 含んでいた．VCard は部品を統括する役割も持っており，利用元部品を含めると機能群として認識できた．

5　考察と今後の課題

実験では，実際のソフトウェアを対象として部品間の利用先部品，利用元部品の集合を作成し，それらの類似度を元にクラスター分析を行った．また，分類した結果が類似しているか調査するために 6 つの基準で部品間の関係を判断した．実験の結果からは，利用先部品が多く一致している集合は，部品同士の役割が同じであったり，機能群としてまとまりを確認でき，関連性を強く認識できる部品の集合であることが確認できた．得られた集合は，その他の基準も満たしていることが多く，何を利用しているかについての情報は，ソフトウェア部品を分類する上で有力な情報であると思われる．一方で利用元部品による類似度の分類では，利用先部品の場合と比べると関連性を確認しづらかったが，分類した部品群に共通の利用元部品を加えると 1 つの役割の一部を実現する部品群になり，意味付けが行えることを確認した．一見すると関連してない部品であってもセットで用いられることがあるような部品群を抽出できると考えられる．

今後の課題として，他のプロジェクトに対して適用を行い，同じような結果が得られるかを確認するとともにクラスターを求める際に，どの部分で区切るのがよいかを検証する必要がある．現在は，類似度が 0.5 以上の部品がクラスターに含まれるように調整しているが，その場合全体の 3 割程度の部品しか分類の対象とならない．類似度に関する閾値を変更することで，分類対象とする部品数がどれくらい含まれるか，分類の対象となる部品の関連性の強さがどのように変わるかを調査し，多くの場合で最適な閾値に基づいて分類ができるようにしたい．

6　まとめ

本研究では，ソフトウェアの利用関係から各部品の類似度を計算し，その類似度を元にソフトウェアを分類することで，似たような部品を抽出する分類手法を提案した．提案した分類方法により，利用関係から関係の高い部品の集合を抽出でき，同じを働きをする部品群に分けることができた．本手法を用いることで，ソフトウェアの部品構成を理解する際に，ひとまとめに理解することが可能な部品群を抽出することができると考えられる．

参考文献

[1] 橋倉大樹, 河合一憲, 近藤貴稔, “利用関係とコードクローン関係に基づく類似部品の分類手法の提案”, 南山大学 情報理工学部 2015 年度卒業論文, 2016.

[2] C. Krueger, “Software Reuse”, ACM Computing Surveys, vol. 24, no. 2, pp. 131-183, 1992.

[3] Classycle : http://classycle.sourceforge.net/.

[4] K. Kobori, T. Yamamoto, M. Matsushita, and K. Inoue.: “Classification of Java Programs in SPARS-J”,International Workshop on Community-Driven Evolution of Knowledge Artifact, 2003.

[5] K. Verco, and M. Wise,:“YAP3 : Improved detection of similarities in computer program and other texts”, Proc. of the 27th SIGCSE Technical Symposium on Computer Science Education, 1996

[6] Art Of Illusion:https://sourceforge.net/projects/aoi/.

[7] Turtle Sport : http://turtlesport.sourceforge.net/.

[8] GeoAPI : http://geoapi.org/.

[9] Pixelitor : http://pixelitor.sourceforge.net/.

[10] jlgui : htttp://sourceforge.net/projects/jlgui/.

[11] Card Me:https://sourceforge.net/projects/cardme/.