利用関係とコードクローン関係に関するメトリクスに基づく類似部品抽出手法の提案

2013SE027 濱島直輝　2013SE103 日下光
指導教員：横森　励士

1　はじめに

近年の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，ソフトウェアを構成する部品を，類似性などを利用して効果的に把握することが求められている．橋倉らは，散布図上で特徴的な場所に配置された部品の集合が，それぞれ同じような役割や性質となっていることを示した [1] が，この方法の場合，特徴的な点で表現される限定された部品群のみが抽出可能である．本研究では，各部品の利用関係とコードクローン関係に関するメトリクスを用いて，非階層クラスター分析による分類を行う．この方法の場合，特徴的な値を持つ集合だけではなく，大きなメトリクスの値を持つ部品全体を分類対象とできると考えられる．実際のプロジェクトに対して提案手法を適用し，分類されたクラスターがどのような構成となっているか，どれくらいの部品が意味のある分類結果となるか，どのような点から類似しているかを調査する．提案手法を用いてソフトウェアを構成する部品を分類することで，保守作業においてどの部品がまとめて把握できるかなどを示すことで，ソフトウェアを構成する部品を効率的に把握することを支援できると考える．

2　関連研究

2.1　ソフトウェア部品グラフ

ソフトウェア部品とは，その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，それぞれのソースコードが記述されてい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ファイルを表し，辺はコードクローン関係や利用関係などの部品間の関係を表現する．本研究では，部品グラフ上の各頂点から出る辺の数，または各頂点へ入る辺の数を数え，利用関係とコードクローン関係に関するメトリクスとして分類に利用する．

2.2　部品の利用関係やコードクローン関係についてのメトリクスを利用した分類手法について

ソフトウェア間において盗用された部品を検出する仕組みとして，L. Prechelt らは JPlag を開発し実際に盗用の検出に成功した [3]．小堀ら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，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効果的に検出した [4]．これらは複数のソフトウェア間に存在する類似した部品を検出する仕組みである，
一方で橋倉らの研究では，1 つのソフトウェアを構成するそれぞれの部品からメトリクスを抽出して，散布図上で同じような値を持つ部品だけを選ぶことで，同じような構成であったり，同じような役割を持つ部品だけをひとまとめで取り出すことができると考えた [1]．利用関係は，外部とのやり取りで実現している機能の関連性を利用している一方で，コードクローン関係は部品内の記述の関連性を利用しており，それぞれのメトリクスは異なる基準で分類を行っていると考えられる．同じような値を示す部品ごとにまとめることで，より類似した特徴を持つ部品をまとめたり，異なる特徴を示す部品に分類する方法を提案した．実際に一致している部分がどれだけ存在するかを，散布図や相関図を使用して関連性を調べることで，部品の理解に必要な情報を提供できることを確認した．

3　利用関係とコードクローン関係に関するメトリクスに基づく類似部品抽出手法の提案

3.1　研究の動機

橋倉らは [1] で，散布図上で部品間の関連性を調査した．この方法の場合，抽出できるのは特徴的な点で表現される限定された部品群のみである．本研究では非階層クラスター分析による分類を行うことで，特徴的な点として集まっている部品だけではなく，分類の基準となるメトリクスの値が大きい部品全体を分類対象とできると考える．また，クラスター分析の基準となるメトリクスの種類を増やすことで，より細かく分類された結果を得ることができると考える．このようにメトリクスの値の観点から似た特徴を持つ部品を効果的に抽出することで，ソフトウェアを効率的に把握することを支援できるのではないかと考えた．

3.2　分析手順

分析手順を以下に示す．

1. 調べたいソフトウェアを CCFinder[5] で解析し，コードクローン情報を取得する．
2. 調べたいソフトウェアを Classycle[6] で解析し，利用関係を取得する．
3. 入手した利用関係，コードクローン関係を CRV[1] に読み込み，部品グラフを構築する．
4. 部品グラフ上の頂点ごとに，利用関係の出力辺数，入力辺数，コードクローン辺数などの情報を求める．さ

らに，分析に利用するメトリクスを選択し，各部品のメトリクス値を表に出力する．

5. 出力された表を入力とし，R 上で非階層クラスター分析を行い，各部品が所属するクラスターを得る．
6. それぞれのクラスターに含まれる部品を分類し，特徴を調査する．分類したそれぞれの部品群の多くの場合，各クラスターには複数の部品群が含まれいることを想定しており，それぞれの部品群を調査対象とする．

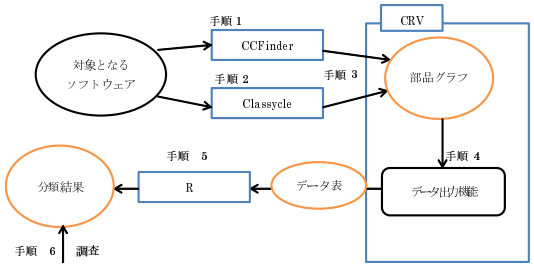

図 1　分析手順

分析では，以下に示す利用関係とコードクローン関係に関するメトリクスを利用する．それぞれのメトリクスが異なった基準で部品を分類しており，複数を組み合わせることで分類結果をさらに分類でき，それぞれの集合が関連性を持つ部品群となると考える．

C Sharing Files　コードクローンを共有するファイルの数を表す．同じクローン集合に属していると同じ値になりやすく，同じ記述を含むことで，似た機能を実現した部品集合になると考えられる．

Uses Internal　利用しているソフトウェア内部のファイル数を表す．同じような機能を利用していると利用先の数も似た値になりやすく，似た機能を利用した部品集合になると考えられる．

Used By　利用されているファイル数を表す．同じような場所から利用されていると似た値になりやすく，同時に使われやすい機能群となると考えられる．

4　適用例

6 つのオープンソースプロジェクトに対し，提案手法を用いて分類した部品群がどのような部品で構成されているのかを調査した．C Sharing Files と Uses Internal の 2 つのメトリクスを用いて分類を行った．部品内の記述の類似性と部品間のやり取りの関連性の 2 つ観点から分類を行うことで，より似た機能を持つ部品の集合を抽出することができ，細かい分類を行うことができると考える．

4.1　適用例における評価項目

以下の調査項目を設定し分類結果を評価する．

Q1　クラスター内の部品群を意味付けできるか．

クラスターをさらに分類した結果である部品群について，下記に示す条件で共通性を見出すことができるかを調査する．条件 1 から条件 4 のうち，1 つでも満たすものをグループの中の部品が類似したものであるとする．部品群の中に，これらの条件の性質を持つ部品群があるかを調査し，類似性を確認することで，クラスター分析で分類を行う手法が有効であるかを示す．

条件 1　同じパッケージに属している．この条件は，開発者が分類した同じ目的を持つ部品のグループであると考えられる．

条件 2　部品群を 1 つのグループとして見たときに機能群を構成している．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点で類似していると判断する．この条件を満たす部品群をまとめて提示することで，機能のかたまりをひとまとめで把握することを支援できる．

条件 3　部品単体が担当している役割が同じである．分類されたグループが特定の役割を果たす部品でまとまり，同様の処理が行われている部品であるという点で関連していると判断する．条件を満たす部品群をまとめて提示することで，どのような処理を行っているのかの把握を支援できる．

条件 4　部品が扱う対象の 7 割以上が同じである．共通の利用先を持つことで条件 2 のように機能群として認識できる部品群となるかを判断する．これにより，同じ役割を持つ機能群を構成している部品群の抽出の支援が期待できる．

また，まとめられる部品がなく，部品 1 つで抽出されたとするグループの部品は

条件 A　部品を統括する部品や，多くの部品に扱われるような，プログラムにおいて重要なファイルである．この条件を満たす部品を提示することで，プログラム全体を把握する際の起点となる部品を示すことができる．

という条件を用いて単体として捉える意味がある部品かどうかを検証する．

Q2　クラスター分析において，特徴が似た部品同士でまとまる部品群を持つクラスターはどれだけ存在するか．

メトリクスの値の小さい部品は，内部の構造が似ていなくても同じ値をとる場合が多いと考えられ，有効な結果はあまり期待できない．一方，メトリクスの値がある程度大きい部品を含むときに有効な結果となると考えられる．どの程度のクラスター，もしくはの部品群が有効な分類結果となるかを調査する．

Q3　プログラムを構成する部品のうち，どれだけの部品が意味付けできるか．

意味付けできた部品を条件 1 から条件 4 のいずれか，もしくは条件 A を満たした部品群に属する部品とする．意味付けができた部品の数を求めることで，提案

手法によりどれだけの割合の部品が分類の対象となるかを調査する．

Q4 非階層クラスター分析を適用する際に，クラスター数を変動すると結果はどう変わるか．
クラスター分析においてクラスター数を変更してそれぞれの条件で得られた結果を比較する．これにより，適切な結果を得るためにどの程度まで分割すべきか，どういう方針で分割数を決めるべきかを調査する．

Q5 クラスター分析において分類に用いるメトリクスの数を増やすことで，より細かく分類できるか．
分類に利用するメトリクスの数を 2 から 3 に増やしたときに，分類結果がどう変化するかを調べる．ここまで用いていた 2 つのメトリクスに，Used By のメトリクスを追加する．3 つのメトリクスを組み合わせることで，同じような部品から利用されている部品ごとに分けることができ，さらに細かい分類が行えると考える．メトリクスの数を増やしたときに，細かく分類した結果を得るために考慮すべきことを考察する．

4.2 適用例における評価結果

A1 Q1 で示した，分析対象としたプロジェクトごとの分析結果を表 1 に示す．表 1 では，プロジェクトごとにクラス数とクラスターとして分割した数，クラスターから部品群への分割が可能で調査対象となったクラスターの数，それらのクラスターから抽出された部品群もしくは単独部品の数，各条件を満たす部品群 (単独部品) の数を示す．

1. 条件 2 と条件 3 のどちらかの条件を満たすグループが多く，両方の条件を満たすグループは存在しない．条件 2 と条件 3 の両方が満たされるということは，部品群として果たす役割も部品そのものの役割も似ているということになり同じ部品となってしまうため，存在しにくい．また，条件 2 を満たしている場合，条件 4 を満たすことが多い．機能群を構成するような部品は，多くの同じ対象を扱っていることが確認できた．
2. C Sharing Files の値が大きいグループは条件 3 を満たすことが多く，Uses Internal の値が大きいグループは条件 2 を満たすことが多い．C Sharing Files は部品内の記述に類似している部分があるため，部品単体での機能が似ており，ある機能を実現する部品群での担当する役割が似てくると考えられる．Uses Internal の値が大きい場合，多くの利用先を持っているので，プログラムの中心的な機能を実現していると考えられる．
3. どちらかのメトリクス値の大きい部品は，各機能の中心となる単体の部品となる事例が多く見られた．また，C Sharing Files と Uses Internal の 2 つのメトリクス値が大きい場合は，部品を統括するような重要な部品が抽出される．

A2 表 1 からは，分割したクラスターのうち，5 割から 7 割程度が意味付けの対象とすることができることが分かった．分類対象となるクラスターはいずれかの値が大きい部品の集合となっていて，それらは 2 から 3 の部品群または単独の部品に分割可能な場合が多い．残りのクラスターはメトリクスの値が小さい部品がまとまってクラスターとなっており，様々な部品が混在している．想定通り，メトリクスの値が大きいクラスターの多くに意味付けすることは可能だが，そうでないクラスターは難しいということが分かった．

A3 A1 で調査した 6 つのプログラムで意味付けできた部品は，部品全体の 1 割から 2 割であった．メトリクスの値が大きいクラスターに属する部品の数は多くなく，多くの部品がメトリクスの値が小さいクラスターに含まれていることが，その原因と考えられる．

A4 Turtle Sport[7] という，GPS と通信をして移動した結果を用いて図表や地図を作成するソフトウェアを分析対象とした場合に C Sharing Files と Uses Internal の指標を用いて分割した場合の結果を評価する．分析対象のソフトウェアは 386 のソースファイルから構成されており，10，15，20，25，30 のクラスター数で分割を行う．クラスターの数が 10 と 15 の場合，ある程度分類可能であったが，1 つのクラスター内に多くの部品群が含まれていた．クラスターの数を 20 とすることで，クラスターの数が 10 と 15 の場合に 1 つのクラスターに混在していた部品群を細かく分類することができた．クラスターの数 25 の場合，分析対象となるクラスターの中身はあまり変化せず，分析対象となりにくいメトリクス値の小さい部品群が細かく分類された．クラスターの数 30 の場合では，意味のある部品群も分かれて，別々のクラスターとして分類された．以上の結果から，メトリクスの値が小さいクラスターによってクラスター数の細かい違いが吸収されてしまうこともあり，分割する数には適切な範囲が存在することが分かる．しかし分割する数の調整だけで良い結果を得ることには限界があり，1 つのクラスターの中に複数のグループが存在することは改善できない．対象のプロジェクトにおいては，クラスターの数は 20 から 25 が最適であると考えられる．

A5 C Sharing Files, Uses Internal, Used By の 3 つの指標を組み合わせてクラスター分析を行った．図 2 は，C Sharing Files と Uses Internal の 2 つをメトリクスとしてクラスター数 20 で分割したあるクラスターに属する部品が，Used By のメトリクスを追加したときにどのように分類されたのかを示した図である．クラスターの数が 20 の場合，ファイルは全て同じクラスターに分類された．このクラスター内には複数の部品のまとまりが混在しており，効果的な分類ができな

表 1　分析対象ごとの評価結果

プロジェクト		クラスター		複数の部品からなるグループ					単独部品に分けられた部品		
プロジェクト名	クラス総数	分割した数	調査対象となったクラスター数	抽出できたグループの数	条件 1	条件 2	条件 3	条件 4	部品数	条件 A	意味付けできた部品数
Turtle Sport	386	20	13	22	18	12	8	17	21	4	69
Art of illution	190	20	10	7	5	1	4	5	10	6	21
Pixelitor	103	10	7	4	4	0	2	3	8	4	12
GeoAPI	257	20	12	7	3	3	4	7	14	2	19
JlGui	70	10	7	4	3	2	2	4	13	5	15
Card Me	118	10	7	2	1	0	2	1	7	4	10

かった．これは 3 つのメトリクスの値を平均したときに，1 つあたりの分割数が減ることによると考えられる．クラスターの数を 30 に増やしたところ，それらの部品群が別々のクラスターになり，より細かい分類に成功した．メトリクスが 2 つのとき，データを管理しているファイルとパネルの表示に関するファイルが混在していたが，メトリクスを増やしクラスターの数を 30 に増やすことで，それらを分類することができた．分類に利用するメトリクスを 3 つに増やす場合，クラスターの数も合わせて増やすことで，より細かく分類することができると考えられる．

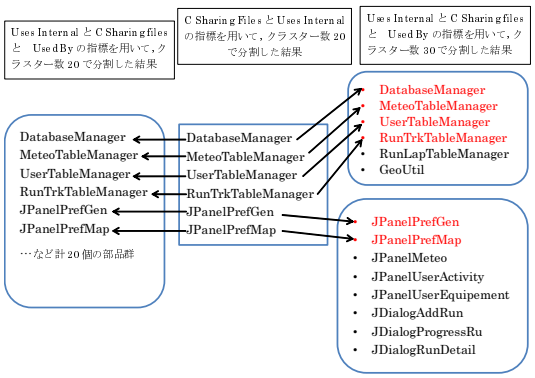

図 2　2 指標の場合と (中)3 指標の場合の結果の違い
(左) クラスター数 20 の場合，(右) クラスター数 30 の場合

5　考察

提案手法を用いてソフトウェア部品を分類した場合，分割したクラスター内には複数のグループが存在し，クラスターの中を改めて分類する必要があることがわかった．最終的に，意味付けを行うことができた部品は全体の 1 割から 2 割で，それらの部品は主に分類対象となるメトリクスの値のいずれか大きい部品であった．単体となった部品は各機能の中心となるような部品となり，複数の部品で構成されるグループは多くの場合で，何らかの共通性を確認できた．提案手法はメトリクスの面から特徴が似ている部品を大まかに分類することで，ひとまとめにして把握することができるような部品群を提示することを支援できると考えられるが，今後の課題としてクラスター内の部品を自動的にどのように分類するかを考える必要がある．

6　まとめ

本研究では，利用関係とコードクローン関係に関する指標を用いて，R 上で非階層クラスター分析を行って部品を抽出する手法を提案した．提案手法により，メトリクスの値がある程度大きいクラスターの中に，各機能の中心となるような単独の部品や何らかの共通性を持つ複数の部品からなるグループが存在することを確認した．提案した抽出方法に基づいて得られる結果を用いて，中心的な機能を実現する部品や機能が似た部品についての情報を提示することで，ソフトウェアを効率的に把握することを支援することができる．今後は，事例を増やし調査することで一般性を持っているかを確認する．また，使用するメトリクスの組み合わせを変えたりメトリクス数を増やすことで，違った特徴を持つ分類結果が得られるかを調査する．

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] 亀井雄佑，木下裕太郎，前原一幾 ：“バージョン間の利用関係の変化を提示するシステムの試作”，南山大学情報理工学部 2012 年度卒業論文，2013.

[3] L. Prechelt, G. Malpohl, M. Philippsen “Jplag : Finding plagiarisms among a set of programs”, Technical Report 2000-1, Fakultat fur Informatik, Universitat Karlsruhe, 2000.

[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 Artifacts, 2003.

[5] T. Kamiya, S. Kusumoto, K. Inoue :“CCFinder : A multilinguistic token-based code clone detection-system for large scale source code”, IEEE Transactions on SoftwareEngineering, vol. 28, no. 7, pp. 654-670, 2002.

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

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