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

2012SE032 橋倉大樹　2012SE103 河合一憲　2012SE123 近藤貴稔
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，ソフトウェアの全体像を効率的に把握することが求められている．利用関係やコードクローン関係などの指標を用いて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一まとめで取り出すことができると考えたが，それぞれの指標が表現するものは微妙に異なっている．扱う情報の異なるメトリクスを組み合わせて分類を行うことで，より類似した特徴を持つ部品のみをまとめたり，異なる特徴を示す部品に分ける際に有効であると考えられる．

本研究では，このように利用関係やコードクローン関係から得られる指標に基づいて，複数の指標を用いて特徴を持つ部品を抽出する方法を提案する．提案する手法に基づいて，メトリクスから得た指標を用いて分類を行い，分類された部品の類似性を確認することで分類結果に対して意味づけを行えるようなツールを作成した．作成したツールを実際のソフトウェアに対して適用し，提案手法の考察を行う．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，ソフトウェアの構成を理解するために有効な部品のまとまりについての必要な情報を提供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部品間の利用関係

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 [3]．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ことができ，Classycle[*1]を利用し部品間の利用関係を抽出する．クラスの継承，インターフェース及び抽象クラスの実装，変数宣言及び，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みなし，部品グラフ上で表現する．部品グラフ上の頂点は各部品を表し，辺は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　コードクローン

コードクローンとは，既存のソースコードの一部を複製することなどによってつくられたコード断片の集合をさす [1]．コピーペーストなどを行うことで発生するが，元のコードに不具合が存在した場合そのすべてのコードクローンに対して修正が必要か検討することが必要となるように，保守性を低下させる要因の一つとして考えられている．本研究では，CCfinder[1] を利用してコードクローン関係を抽出し，ある一定以上の長さで連続して一致しているコード片が複数存在する場合にコードクローン関係とみなす．部品グラフ上では頂点を各部品で表し．類似したコード片を共有する部品同士を無向辺で表現する．

2.3　分析手法について

コードクローンの分析環境として，ICCA[2], CloneWarrior[*2] などが提案されている．これらのツールでは，コードクローンに関する情報を CCfinder から入手したうえでコードクローンに関連するメトリクスを入手し，それに基づいてコードクローン分析を行う．過去の研究では，ソフトウェアの内部の構造がどのように複雑化するかを調べるために，バージョン毎の部品の利用関係の変化を視覚化するツール URV(Use Relation Viewer)[4] などが提案されてきた．これらのツールでは，バージョン毎の分析を行い，部品グラフ上で書される各クラスの利用関係やコードクローン関係の数がどのように変化したかを表やグラフを用いて表示する．

3　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方法の提案

3.1　研究の動機

利用関係やコードクローン関係などの指標を用いて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一まとめで取り出すことができると考えられる．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が，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いる．このように，分類を行う際にそれぞれのメトリクスによって注目している情報が異なるので，分類に利用するメトリクスによって得られる結果は異なると考えられる．本研究では，複数の観点から分類を行い，二つの特徴が同じ特徴を示す部品ごとにまとめることで，より強い類似性を示す部品群を抽出することを目的としている．注目する情報の異なるメトリクスを組み合わせて分類を行うことで，

[*1] Classycle : http://classycle.sourceforge.net/

[*2] CloneWarrior : http://se-naist.jp/old/clonewarrior/

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ける際に有効であると考えられる．ただし，分類から得た結果だけでは，分類結果に意味があると判断することができない．よって，実際に分類された特徴毎に対して意味づけを行っていくことで，その特徴を示す部品のまとまりが担う共通の役割を具体的に確認できる．

3.2 分類の手順

分類の基準として，利用関係やコードクローンに関するメトリクスを抽出する．現在抽出しているメトリクスは次の表 1 のようになっている．これらの基準を組み合わせて分類し，それぞれの抽出できたグループに対して意味づけを行い，それぞれの特徴を持つグループの部品の役割を把握することで，部品の観点からソフトウェアの全体像の理解を支援する．

1. それぞれのメトリクスから組み合わせて分類を行うための指標を二つ選択する．
2. 選択した二つの指標の観点から分類を行い，二つの特徴が同じような特徴を示す部品ごとにまとめる．
3. 同じ特徴を示した部品ごとに，実際に一致している部品がどれだけ存在するかを確認する．

4 実装

4.1 CRV

本研究では，提案する分類手法のそれぞれの手順において必要な情報を提供するような機能を実装した CRV（Component Relation Viewer）を実現した．CRV では，表などを用いて各部品単位で各指標の値を表示する機能，散布図を用いて部品の分類ができるような簡易機能を実装した．その部品に関連した利用関係やコードクローンが類似しているかなどを視覚的に評価できる機能を作成した．

4.2 CRV の分析の手順

ツールの分析手順は以下のようになっている．

1. 調べたいソフトウェアを CCFinder で解析し，クローン情報を示したファイルを取得する．
2. 調べたいソフトウェアを Classycle で解析し，利用関係を表した XML ファイルを取得する．
3. 1，2 の結果を読み込み，データベースに格納する．
4. データベースに格納されたファイル間の関係情報をもとに表を作成し，ユーザーに提示する．
5. ユーザーの要求に応じて，表のソートや検索，散布図や相関図の作成を行う．また，ユーザーの指定に応じて新しい表やグラフを作成し，分析の補助を行う．

4.3 結果の表示，分類に関する機能

表の表示：手順 1 でそれぞれのメトリクスから組み合わせる指標を選ぶためにはそれぞれの指標の分布の情報が必要がある．そこで，データベースに格納された情報をファイル単位で表示し，調査対象のソフトウェアの部品の一覧を表示する機能を作成した．指定した要素をソートしたり，表からファイル名を検索することで，各指標がどのように分布しているのか確認できる．

散布図：手順 2 で選択した二つの指標の観点から分類を行い，二つの特徴が同じ特徴を持つ部品をまとめるためには，二つの指標の値から同じ特徴を示すよう部品に分類するための情報が必要である．そこで，利用者がコードクローン関係と利用関係の要素を選び X 軸と Y 軸の値とした散布図を作成する機能を作成した．利用者は閾値の数を指定した上で，それぞれのセルに配置されるファイルの情報を入手できる，二つの指標の値が似ている部品の集合ごとに部品を抽出することで二つの指標の値が似ている部品の集合ごとに部品を抽出することで，同じ特徴を示すような部品ごとに分類することができる．

4.4 関連のある部品の表示機能

相関図： 同じ特徴を示した部品ごとに，一致している部品がどれだけ存在するかを確認するためはグループごとにそれぞれの部品手順 3 で同士の共通性が確認できる情報が必要である．そこで，分析対象のファイルを指定し，どのファイルとクローン関係や利用関係があるかを示した相関図を表示する機能を作成した．相関図では分析対象のファイルを中心に配置し，その周囲に関係を持つファイルを配置して，二つの間に線を引いた図を作成し分析する．図 1 は相関図の例である．指定された二つのファイルについて相関図を作成し，周囲に配置されるファイルのファイル ID と関係が一致した場合には強調して表示する．

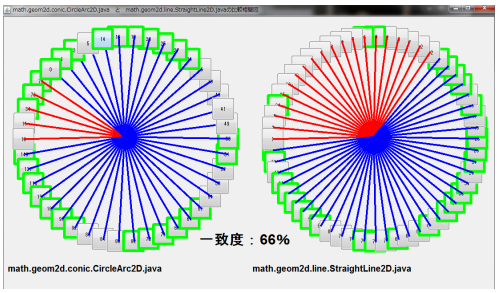

図 1 相関図による比較の例

5 分類手法の適用例

一つの指標から得られた分類結果にもう一つの指標を組み合わせることで，更に詳細な分類を行うことができることを確認するために適用例を示す．一つの指標による分類を行うことで得た分類結果にもう一つの指標を組み合わせて分類を行うことで，より似た特徴を示す部品群に分類され相関図を用いてそれらに意味づけをすることで，各特徴ごとの強い類似性を示すことができる．

表 1　CCfinder と Classycle から抽出したメトリクス

要素名	要素の説明	期待できる分類結果
コードクローンを共有するファイルの数	クローン集合におけるファイルの出現回数	類似した値を持つ部品群に対して，類似した機能を持つ部品群に分類できると考えられる
コードクローンセットの保持数	クローン集合におけるファイルの出現回数	コードクローンとなるコード片を何種類持っているかという情報から，他の部品のソースコードと一致度の高い部品を分類できると考えられる
コードクローンとなっているコード片の数	クローン集合におけるファイルの出現回数（のべ）	既存のコードをコピーした回数から，ソフトウェア保守が行き届いていない部品を分類できると考えられる
被利用関係を持つファイルの数	そのファイルを利用しているソースコードファイルの数	類似した部品群から利用され，同一の機能を実現するためにデータを受け渡したり，データの処理を行う部品群に分類できると考えられる
利用内部ファイル数	そのファイルが利用しているソフトウェア内部のソースコードファイルの数	類似したファイルを利用し，同様の機能を実現している部品群に分類できると考えられる
利用外部ファイル数	そのファイルが利用している外部ライブラリクラスの数	類似した外部ライブラリを利用し，類似したデータの処理を行っている部品群に分類できると考えられる

5.1　適用例

まず，表を用いて分類を行い，組み合わせて分類をするための指標の選択を行う (手順 1)．図 2 は CardMe の部品をコードクローンを共有する数の降順にソートした表の一部で，上位の約 30 ファイルはそれぞれ約 40 ファイルとコードクローンを共有していることが分かった．同程度のコードクローンを共有しているファイルが多く存在することから，これらの部品は似た特徴をもつ可能性があると考えられる．また，図 2 からこれらの部品において利用内部ファイルを多く持つ部品があることから，この 2 つの指標を選択して分類を行う．

ClassName	CodeClone	CloneSet	Uses Internal
net.sourceforge.cardme.vcard.types.AdrType.java	45	29	11
net.sourceforge.cardme.vcard.VCardImpl.java	39	75	43
net.sourceforge.cardme.vcard.types.ProdIdType.java	38	15	9
net.sourceforge.cardme.vcard.types.SoundType.java	38	14	11
net.sourceforge.cardme.vcard.types.RevType.java	38	13	9
net.sourceforge.cardme.vcard.types.ProfileType.java	38	16	9

図 2　コードクローンを共有する数の降順にソートした表

コードクローンを共有するファイルの数を横軸，利用内部ファイル数を縦軸の値とした散布図を利用して，これらの指標の観点から分類を行い 2 つの特徴が同じような特徴を示す部品をまとめる (手順 2)．図 3 は，実際に適用し生成されたテキストファイルの情報を元に作成した散布図であり，それぞれごとに類似性を示すことで以下の特徴を確認できた (手順 3)．

(ア)　利用内部ファイル数が多くないがコードクローンを共有するファイルの数が多い特徴を持つ部品をまとめることができる．図 4 は vcard.types パッケージに属する指定したファイルを対象に作成した相関図をいくつか表示したものである．このグループの相関図を調べたところ，丸で囲んだファイルとその関係がすべてのグループ内の部品の相関図に共通して現れることを確認できた．このことから，vcard.types パッケージは，ほぼ共通の役割を持った部品の集合だと分かる．

(イ)　利用内部ファイル数が多く，コードクローンを共有するファイルの数が多い特徴を示した

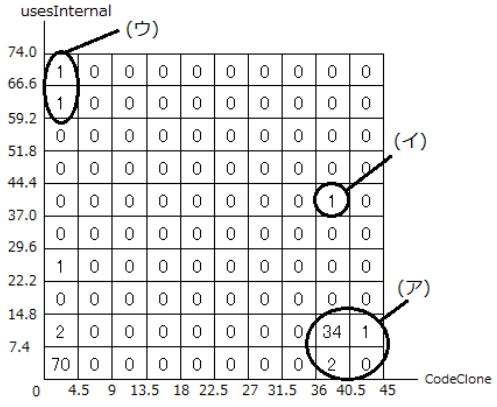

図 3　CardMe をモデル化した散布図

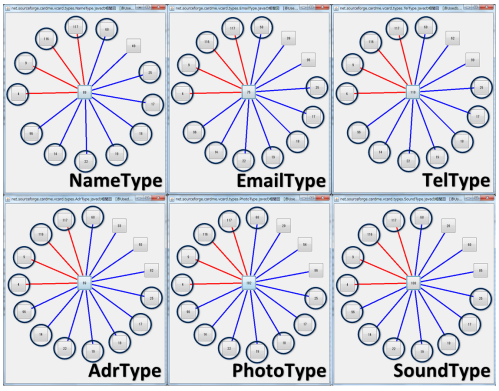

図 4　vcard.types パッケージに属する部品の相関図

vcard.VCardImpl が存在する．この部品はコードクローンを共有するファイルの数だけで分類すれば（ア）の部品群と同じように分類される．しかし，そこから利用内部ファイル数で分類することで（ア）からさらに分類することができた．図 5 は，（ア）の部品群の中の部品の 1 つである vcard.types.PhotoType と vcard.VCardImpl のソフトウェア内部の利用関係に関する相関図を作成し実際に比較した図である．共通して現れるファイルは非常に少なく一致度も 13% と小さいため，vcard.VCardImpl は（ア）の部品群とは役割が違う部品であることを確認できた．また，全て

の vcard.type パッケージのファイルを利用していることから，vcard.VCardImpl は vcard.types パッケージと深く関係し，機能面ではこれらの部品を管理する重要なファイルであると考えられる．

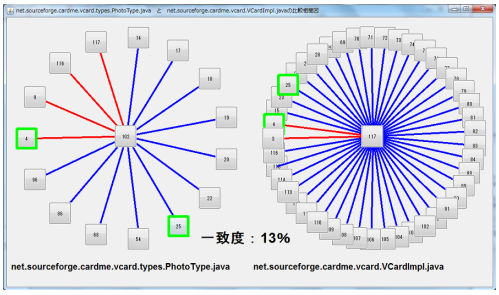

図 5 vcard.types.PhotoType と vcard.VCardImpl の比較

(ウ) コードクローンを共有するファイルの数は少ないが利用内部ファイル数が多く特徴を示した engine.VCardEngine と io.VCardWriter がまとめられた．これらの部品はソフトウェア内の半分以上のファイルを利用している．共通して現れるファイルが多く一致度も 86% と多くの関係のある部品が一致していることを確認でき，この二つの部品は類似した役割を持つ部品であると考えられた．これらの部品は独自のコードを有しており，CardMe において中心的なファイルであることがわかった．

6 考察

6.1 分類に用いる基準について

コードクローン関係，利用関係のそれぞれについて分類に用いる指標を抽出する際に，何通りかの意味の異なる指標が提案できる．図 2 で示すように同じ部品間の関係から抽出されるメトリクスでも異なる意味を持ち，それぞれの分類結果が異なる．

指標を組み合わせた分類を行うことで，より類似した特徴を持つ部品をまとめ，異なる部品と分けるられることも確認できた．適用例における (ア) のように，それぞれの指標の特徴において似た特徴を示した部品を抽出することで，類似した役割を持つ部品ごとにまとめることができた．適用例の（イ）は，コードクローンを共有する部品の数では（ア）のグループと同じ特徴を示した．しかし，利用内部ファイル数でさらに分類を行うことで異なる特徴を示す部品に分類できた．このように 1 つの指標が多くても，もう 1 つの指標の数によって分類結果は異なることから，2 つの指標からより類似した特徴を示す部品の分類と異なる部品の分類を行うことができる．

組み合わせる指標を変えて分類することで別の分類結果となることを示すことも確認できた．適用例と同じソフトウェアに対し，コードクローンを共有するファイルと被利用関係を持つファイルから分類を行ったところ，適用例とは異なり vcard.arch パッケージを多く持つ部品群をまとめることができた．このように組み合わせる指標を変えて分類を行うことで分類結果が異なることから，2 つの指標から分類を行う有利性を示すことができたと考える．

6.2 機能の拡張の方向性

表や散布図を用いることで，一つの指標による分類からは確認できなかった特徴のある部品を抽出したり，二つの特徴が同じ特徴を示す部品ごとにまとめることができる事を確認した．本研究にて作成した CRV では，部品の分類を行う機能として表の表示や散布図の作成など最低限の機能のみを実装しており，分類を行う際に R や SPSS，Excel などの既存の統計処理ツールを外部ツールとして用いることで，部品の分類をより効果的に行えると考える．

相関図を用いることで，同じ特徴を示した部品ごとに，実際に一致している部分がどれだけ存在するかを確認することで，分類されたグループ毎に対して意味づけを行うことができた．今後は，調査したい部品と他の部品が持つ関係のある部品がどれだけ一致しているのかをすべて調べ，関係のある部品の一致度のランキングを作成する機能を実装することで，分類されたグループ全体の類似性を視覚的に把握することができると考えられる．

7 まとめ

本研究では，コードクローン関係と利用関係の複数の観点から分類を行い，二つの特徴が同じ特徴を示す部品ごとにまとめることで，それらの部品により強い類似性を示す分類方法を提案した．提案した分類方法により，コードクローン関係と利用関係から部品を分類することができ，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づけが行えることを確認した．提案した分類方法に基づいてソフトウェア部品の分類を行うことで，部品からの観点からソフトウェアの構成を理解するための情報の提供を支援することができる．

参考文献

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

[2] Y. Higo, N. Yoshida, T. Kamiya, S. Kusumoto, K. Inoue : “Code Clone Analysis Tool : ICCA, ” The Second International Workshop on Biologically Inspired Approaches to Advanced Information, 2006.

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

[4] 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.