Vol.2016-SE-194 No.8
2016/11/18

部品間の関係を利用したソフトウェア部品の分類手法の提案

横森 励士[1,a)]

概要：求められる機能の増大や保守期間の長期化に伴い，近年のソフトウェアは大規模なものとなっている．ソフトウェアを構成する部品それぞれが実現する機能の類似性や，部品が担っている役割の類似性などを利用することで，大規模化したソフトウェアを効果的に把握することができるようになると考える．本研究では，あるソフトウェアを構成する部品群から利用関係やコードクローン関係を抽出し，それらの関係の類似性から機能や役割が類似している部品を抽出する手法を提案する．分類の方法として，部品グラフ上での各部品の入出力辺の数を用いて値の似ている部品ごとに分類する方法，利用先（利用元）の部品がどれだけ一致しているかから各部品間の距離を求めて分類する方法の二通りの方法を提案し，評価実験からそれぞれの方法の特徴を考察する．

キーワード：ソフトウェア部品，クラスター分析，利用関係，コードクローン関係

Classification Methods for Software Components based on Relationships between Software Components

Abstract: In recent years, the size of software becomes more large scale. So the number of software components that compose one software is also increasing and the relationships between software components becomes more complicated. So we would like to assist software comprehension by presenting components that play similar roles, or components whose positions are almost the same. In this paper, we suggest two classification methods for software components based on relationships between software components. One is based on the similarity of several metrics based on the component graph, and the other is based on the similarity of using components (or components that are used by). We will show application examples of the two classification methods and discuss their characteristics.

Keywords: Software Component, cluster analysis, use-relation, relation about code clone

1. はじめに

近年のソフトウェア開発現場では，開発対象であるソフトウェアの大規模化，複雑化が急激に進行している．それに伴い実際にソフトウェアが保守や機能追加の対象となる期間も長期化している．長期間にわたって開発されるソフトウェアでは，段階的に機能が追加され，様々な仕様変更への対応がなされている．当初はシンプルな構成であったとしても，ソフトウェア内部の関係は次第に複雑化・大規模化していき，プログラムを構成する部品の数が増えることで全体を把握することが難しくなり，さらに部品間の関係も複雑化していくことで，プログラムの内部構造の理解は困難なものとなる．大規模かつ複雑になったソフトウェアを理解するために必要な労力も，同様に増加していると考えられ，ソフトウェアを構成する部品それぞれが実現する機能の類似性や，部品が担っている役割の類似性などを利用することで，ソフトウェアを効率的に把握することができるようになると考える．

本研究では，利用関係やコピーされた部品などの部品間の関係に着目し，それらの関係に基づいてソフトウェア内で同じような性質を持つ部品を部品群に分類することを目的とする．提案手法では，あるソフトウェアを構成する部品群から利用関係やコードクローン関係を抽出し，それら

[1] 南山大学　理工学部
nanzan university, 18 Yamazato-cho, Shouwa-ku, Nagoya, 466-8673, Japan
[a)] yokomori@se.nanzan-u.ac.jp

の関係の類似性から機能や役割が類似している部品を抽出する．分類の方法として，部品グラフ上での各部品の入出力辺の数などを用いて値の似ている部品ごとに分類する方法，利用先（利用元）の部品がどれだけ一致しているかという類似度から各部品間の距離を求めて分類する方法の二通りの方法を提案する．

前者は，部品グラフから直接得られた値を用いて，値の類似性をもとに類似部品を判定する．部品グラフにおける利用関係の入力辺の数，利用関係の出力辺の数，コードクローンを共有している部品の数といった判定要素をいくつか組み合わせて，クラスター分析を行い，値の傾向が似ている部品群を抽出する．それぞれの値自体が弱い関係性を持つと考えられ，違った意味を表す複数の指標を用いることで，効果的に分類ができるのではないかと考えた．

一方で後者は，部品グラフ上で表現される利用関係から，同じ部品を利用している，もしくは同じ部品から利用されているという，より強い関係性を示す指標を用意し，その指標に基づいてクラスター分析を行い，分類を行う．それぞれの部品において，利用している部品や，利用されている部品の集合を求め，各部品対について集合の類似度を求め，部品間の距離に関する距離行列を作成する。

オープンソース開発プロジェクトのソースコードへ本手法の適用を行い，得られた結果から各分類方法の特徴を考察する．分類結果を用いることで，分析対象ソフトウェアを機能的なまとまりとして把握することを支援できるとともに，似たような機能を実現する部品，ある部品で実現している機能群に含まれる部品という形で，ソフトウェアを構成する部品を部品群として把握できると考えられる．

以下では，2 章で背景技術を紹介したのちに，3 章で提案手法における 2 種類の分類手法について概要を紹介する．4 章，5 章でそれぞれの方法の手順や適用事例を紹介する．

2. 背景技術

2.1 ソフトウェア部品と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指す [6]。本研究では，Java で記述された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ているファイルを部品の単位として扱う．

ソフトウェア部品間には，ある部品がある部品を利用する，もしくは，ある部品がある部品と類似コード片を共有しているなどの関係を設定することができる．これらのソフトウェア部品間の関係をグラフ構造を用いてモデル化し，ソフトウェア部品グラフとして表現する．ソフトウェア部品グラフにおける各頂点はそれぞれの部品を表し，辺は部品間の関係を表現する．それぞれの分析手法によって，部品グラフ上で表現する辺が指し示す意味が異なることが考えられるが，本研究では，後述する利用関係の辺，もしくはコードクローン関係の辺，もしくは両方の辺を部品グラフ上で表現する．

2.2 ソフトウェア部品グラフ上で表現する辺について

ソフトウェア部品における利用関係とは，ある部品が自分の機能を実現する際に，他の部品の持つ機能を利用して実現することを指す．これらの関係を理解することで，ある機能が実現している機能が，ソフトウェアのどの部品群を参照することで把握できるかなどを支援できる．本研究では，利用関係を解析するために， Classycle[3] を利用する．利用関係として，クラスの継承，インスタンスの生成，メソッドの呼び出し，フィールドの参照を利用関係として抽出し，利用元の部品から利用先の部品への有向辺として定義し，部品グラフ上で表現する．

ソフトウェア部品においては，類似した記述をコード断片として共有する部品が存在する場合がある．このような部品をまとめて把握しておくことで，ある修正が必要になった場合に同種の対策を行うべきかの検討が必要となる部品の集合を取得することが容易となると考えられる．そのような，類似した記述をコード断片として共有している部品間に，コードクローン関係を表す辺を定義する．本研究では，コードクローン関係を解析するために， CCfinder[5] を利用する．一致するコード片が 30 トークン以上にわたるコード片を共有している部品間にコードクローン関係が存在すると定義する．こちらの関係は，向きを考慮せずに，無向辺として部品グラフ上で表現する．

2.3 ソフトウェア部品の特徴を表すメトリクスを用いた分類手法について

ソフトウェア部品の特徴を表すメトリクスをソースコードから入手して，同一もしくは類似した部品を抽出する手法として，小堀らは Luigi を提案している [11]．この方法では，制御構造の複雑さを表現するサイクロマチック数やメソッド数などを主メトリクスと定義したうえで，規模を表現するやトークン数や主メトリクスが類似しているかに重点を置いたうえで，同一もしくは類似した値の部品を抽出する．この手法は，クラス単位のコピーアンドペーストを捕捉することを主な目的としており，ソフトウェア部品検索システムにおいて多数のソフトウェアから部品を抽出したのちに，各ソフトウェアを構成する部品が接続された状態となるように，ソフトウェア間で同一と思われる部品を抽出する手法である．

今回我々が提案しようとしている手法は，ある一つのソフトウェアの中に存在する，類似した機能を実現している部品や，ソフトウェア内での部品の立場，役割などが類似している部品を部品群としてまとめることを目的としている．

事前研究として，橋倉らは，ソフトウェアを構成するそれぞれの部品からその部品の特徴をあらわすメトリクスを抽出して，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 [12]．メトリクスの例として，部品間の利用関係が挙げられる．内部の記述が違っていても，扱う対象や担う役割が同じであれば，結果としてその部品が利用している部品の数（もしくはその部品を利用している部品の数）は同じような値となりやすい．そこで，利用部品数や被利用部品数，外部ライブラリクラスの利用数などで分類することで，同じような役割の部品をまとめることができると考えた．

他に分類に利用できるメトリクスの例として，コードクローン関係が挙げられる．コードクローン関係とは，共通のコード片の集合を持つ部品間の関係を示したもので，どれだけ他の部品と類似したコード片を共有しているかなどの指標を抽出することができる．多くのファイルが関係する大規模なクローンセットができている場合，そのクローンセットはグラフ上で強連結となっていることが想定される．ファイルごとに，類似コード片を共有しているファイルの数を求めた場合，大規模で強連結となっているクローンセットに含まれる部品は同じような値となりやすいことが想像できる．そこで，その部品と類似したコード片を共有している部品の数で分類することで，コードクローン関係が同じような部品同士をまとめることができると考えた．

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．[12] では，散布図を用いてコードクローン関係と利用関係が得られるメトリクスを抽出し，同じような値を示す部品ごとにまとめることで，より細かく分類する方法を提案した．提案した分類方法により，同じ特徴を示した部品ごとに関連性を調べることで，類似した機能を実現している部品や，ソフトウェア内での部品の立場，役割などが類似している部品という形で，分類した結果に意味付けが行えることを確認した．ただし，橋倉らの手法の場合，散布図上で値の関連性を調査していたので，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．

2.4 コードクローン関係や利用関係を利用した分析手法について

利用関係を利用した解析方法として，アーキテクチャに関する情報をソースコードから取得するための方法が提案されている．Zhang は OO システムをグラフとしてモデル化することで，ソフトウェアアーキテクチャを取得するためのクラスタリングアルゴリズムを提案している [10]．Constantinou はコンポーネント間の階層的な関係をもとに，アーキテクチャのレイヤーとデザインメトリクスとの関係を調査した [4]．

コードクローン関係を利用した解析方法として，Mondal は コードクローンの安定性を種類ごとに調査し，ギャップドクローンと呼ばれる，タイプ 3 のコードクローンが高い安定性を持っていることを確認した [7]．Antoniol は 19 のバージョンのリナックスカーネルにおいて，複製されたコードがどのように変化したかを調査している [1]．吉田らは，クローンを有するコード間の依存関係に基づいてリファクタリングを支援する方法を提案している [9]．

3. ソフトウェア部品の特徴を表すメトリクスを用いた分類手法について

本研究では，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，部品間の関係を利用してソフトウェア部品の分類を行う手法として，2 種類の方法を提案する．

(1) 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分類手法
部品グラフ上で表現されている，利用関係やコードクローン関係の辺の数などのメトリクスを抽出し，値が類似しているという観点から似た特徴を持つ部品を効果的に抽出する．2 つ以上の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とで，特徴的な点として集まっている部品だけではなく，メトリクス値が大きい部品全体を分類対象とできると考えた．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た．

(2) 利用先（利用元）部品の一致度に基づく分類手法
利用関係だけを表した部品グラフを用意し，頂点ごとに利用先部品の集合（各頂点から出る辺の到着先部品の集合）や，利用元部品の集合（各頂点へ入る辺の出発元部品の集合）を求める．それぞれの部品対ごとに，利用先（もしくは利用元）部品の集合がどれだけ一致しているかを類似度として求める．部品間の類似度をもとに階層的クラスター分析を行い，同じ部品を利用している部品，同じ部品から利用されている部品を抽出する．機能的に似ていると考えられる部品群や，ある部品から利用される機能群としてとらえることができるような部品群を抽出できると考えた．

以下の章では，それぞれの手法の分析手順と適用結果を紹介する．

4. 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分類手法

4.1 分析の手順

利用関係やコードクローン関係に関するメトリクスとして表 1 で表すメトリクス群を利用する．これらの指標の中

表 1　使用するメトリクスの一覧

メトリクス名	説明	取得方法	予想される分類結果
CSharingFiles	コードクローンを共有するファイルの数	その頂点から出るコードクローン関係の辺の数	同じクローンセットに入っている部品ごとの分類
Cset	類似コード片の種類数	分析結果内のクローン集合における出現数（複数回出現は 1 回）	類似コード片の種類数による分類
Capperance	類似コード片の出現回数	分析結果内のクローン集合における出現数（延べ数）	類似コード片の出現回数による分類
UsedBy	その部品を利用しているファイルの数	その頂点に入ってくる利用関係の辺の数	同じような部品から利用されている部品の集合に分類
UsesInternal	その部品が利用しているファイルの数	その頂点から出る利用関係の辺の数	類似した機能群を利用している部品の集合に分類
UsesExternal	利用している外部ライブラリクラスの数	分析結果内における各クラスの外部クラスの使用数	類似した外部ライブラリ群を利用している部品の集合に分類

からいくつかを組み合わせて分類する．分類できた部品群ごとに意味付けを行い，それぞれの部品群の特徴を調査する．それぞれの部品群が，それぞれ異なった役割を持つ部品の集合に分かれていることを確認し，分類できたそれぞれの集合を把握することでソフトウェアがどのように構成されているかを理解することを支援する．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，クローン情報を取得する．
(2) 調べたいソフトウェアを Classycle[3] 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表 1 で示す情報を求める．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) 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調査し，各クラスターの特徴を調査する．

4.2　評価実験の概要

実際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．Turtle Sport[8] 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する．Turtle Sport のバージョン 1.0 について，コードクローン関係と利用関係から部品の分類を行う．このソフトウェアのファイル総数は 386 である．

4.3　評価項目

提案手法の有効性を確認するために，以下の調査項目を設定し，分類結果を評価する．

Q1　提案手法を用いて抽出した部品群の中に，事前研究で抽出した部品群と同じような特徴を持つ部品群が存在するか．

事前研究 [12] で，Uses Internal（利用ファイル数）と C Sharing Files（コードクローンを共有するファイルの数）を用いて分類したときの部品群の特徴は

(1) 利用先部品の多くが一致するような，ソフトウェア内で同様の役割を果たすことが認識できるようなファイル群である．
(2) 多くの共通の利用先を持ち，それぞれが機能群の中の一つの機能を実現しているようなファイル群である
(3) 部品を統括する中心的な機能を担うファイルは，単一メトリクスでは他のグループと混ざっていても，複数のメトリクスを用いることで分離できる

というものであった．提案手法を用いて抽出できた部品群の中に，これらの性質を持つ部品群があるかを調査する．これにより，メトリクス値を用いて分類を行う手法が，アプローチが違っても似た性質を持っていることを示す．

Q2 非階層クラスター分析を適応する際にどの程度まで分割した結果を利用すべきか.
クラスター分析において求めるクラスター数を変更して得られた結果を比較する．適切な結果を得るために最適な分割数を調査する.

Q3 分析後の部品群において，特徴が似た部品同士でまとまる部品群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．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．分割後の部品群を調査し，どの程度の部品群が有効な分類結果となっているかを調査する.

Q4 クラスター分析において分類対象とな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.
この項目では，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．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.

4.4 適用結果

A1 Turtle Sport に対し，Uses Internal と C Sharing Files を指標とし，非階層クラスター分析を用いて分類し，20 の部品群に分割した結果を図 1 に示す．点はそれぞれのクラスターの中心の座標を示しており，原点付近にクラスターが多く生成していることが分かる.
利用先部品の多くが一致するような，ソフトウェア内で同様の役割を果たすことが認識できるようなファイル群として，以下のような部品で構成されるグループ 8 が存在した．上 4 つはデータを管理しているファイルであり，ほぼ共通の役割を持った部品の集合である.

{ DatabaseManager，MeteoTableManager，
RunTrkTableManager，UserTableManager，
JPanelPrefGen，JPanelPrefMap

多くの共通の利用先を持ち，それぞれが機能群の中の一つの機能を実現しているようなファイル群として，以下のような部品で構成されるグループ 16 が存在した．上記の 5 つは位置情報に関するファイルで，下記の 3 つは図に関するファイルであった．Turtle Sport は 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で，それらの機能の一つ一つを担っている..

{ HstFile，TcxFile，GoogleMapGeo，
ModelRun，GpxFile，JDiagramComponent，
JDialogPreference，JDialogRunSendEmail

部品を統括するような中心的な機能を担うファイルとして，グループ 5 に MainGui というファイルが存在した．グループ 5 を構成する部品は以下の通りである.
MainGui は部品を統括する重要なファイルであり，二つのメトリクスでは，上記のようにパネルやダイアログに関するファイルと混在し分離できなかった．メトリクスに Used By（利用されているファイルの数）を追加し，三つのメトリクスの組み合わせでクラスター分析を行うことで，MainGui を分離できた.

{ JPanelTableRun，JPanelTreeRun，
JDialogImport，JPanelRun，
JPanelStat，JPanelUserProfile，MainGui

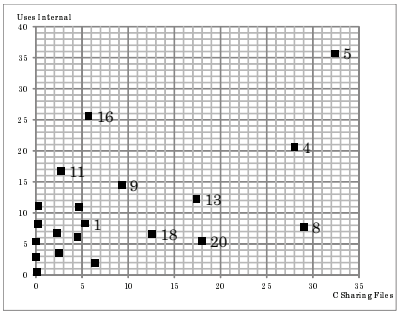

図 1 Uses Internal と C Sharing Files の指標を使用し，非階層クラスター分析をした結果 (クラスター数 20)

A2 A1 と同じ指標で非階層クラスター分析を用いて，クラスターの数を 10，20，30 の部品群でそれぞれ分割した．分割した結果を図 2，図 1，図 3 に示す．クラスターの数が 10 の場合，ある程度までは分類できたが，1 つのグループ内にいくつもの部品群が含まれていた．クラスターの数が 20 の場合，クラスターの数が 10 の場合に 1 つのグループに混在していた部品群をそれぞれの部品群に細かく分割することができた．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，別々のグループとして分類された．以上の結果から，本プロジェクトにおいては，クラスターの数は 20 が適していたと考えられる.

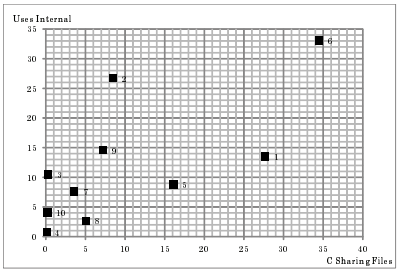

図 2 Uses Internal と C Sharing Files の指標を使用し，非階層クラスター分析をした結果 (クラスター数 10)

A3 クラスターの数を 20 にしたときの分類結果の中で，いくつのグループが似た役割の部品から構成される部品群であったかを調査する．図 1 の 20 の点を大きく分けて，メトリクスの値が大きい値のグループ (図中

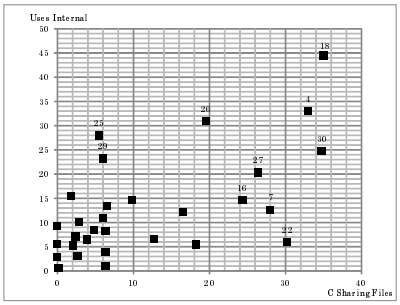

図 3　Uses Internal と C Sharing Files の指標を使用し，非階層クラスター分析をした結果 (クラスター数 30)

で番号付き) と小さい値のグループ (番号なし) に分けた．番号付きのグループにおいては，10 グループのうち，8 つのグループが似た役割の部品から構成される部品群に分割できた．一方で番号なしのグループでは，全てのグループが多種の部品が混在したグループとなっていた．今回の結果から，メトリクスの値が大きいグループの多くに意味付けすることは可能だが，そうでないグループについては難しいということと，また全体として見た場合 3 割から 4 割が意味のあるグループに分類できると考えられる．

A4　C Sharing Files，Uses Internal，Used By の三つの指標を組み合わせてクラスター分析を行った．図 4 は，図 1 のグループ 8 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，グループ 8 のファイルは全て同じグループに分類された．このグループ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 つあたりの重み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グループになり，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，適切に分類することができると考えられる．

4.5　評価実験の考察

事前研究において得られた部品群と同じ性質の部品群を得ることができており，メトリクス値を用いて分類を行うというアプローチは，値の大きい部品に対して有効に働くことを確認した．今後，適用事例を増やし，手法によって得られる知見の一般性を高めたい．さらに，分類に利用しているメトリクスの組み合わせは限定されており，今後は他の組み合わせにおいて得られた分割結果に対する意味付

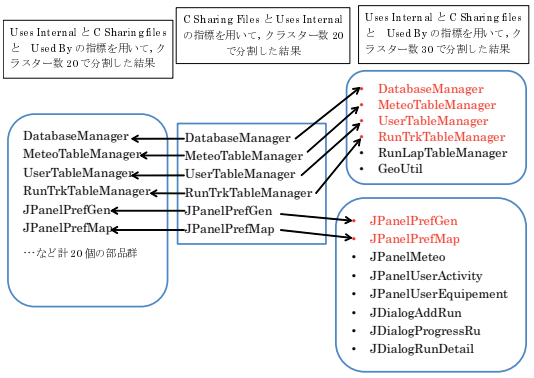

図 4　2 指標の場合と (中)3 指標の場合の結果の違い
(左) クラスター数 20 の場合，(右) クラスター数 30 の場合

けも行っていきたい．

また，実プロジェクトに対して適用を行う際には，それぞれのプロジェクトに対してプロジェクトのサイズ（ファイル数）や利用するメトリクスの種類数などによって決定できるような適切な分割数があるとも考えられ，それらの推測も今後の課題であると考えられる．

5.　利用先部品の一致度に基づく分類手法

5.1　分析の手順

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利用先部品もしくは利用元部品の機能が類似した部品ごとに分類を行うために，次のような手順で利用関係の整理と部品間の類似度の計算を行う．ツールでは部品間の類似度から生成した行列を出力し，出力された行列を距離行列として R に入力し，階層的クラスター分析を用いて分類を行う．R での分析結果として出力された樹形図から分類された部品群を読み取り，それぞれのグループの特徴を確認する．

(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) 利用関係から部品グラフを構築し，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) 利用元部品の類似度，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，各部品対ごとに類似度を計算し，距離へ変換する．
(4) 各部品間の距離を示す距離行列を作成し，階層的クラスター分析を行い樹形図を得る．
(5) 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し，部品の類似性を確認する．

5.2　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 C_A，ある部品 B の利用元 (もしくは利用先) 部品の集合を C_B としたとき，それらの集合間の類似度 $sim(C_A, C_B)$ を

Jaccard 係数を用いて示す．

$$sim(C_A, C_B) = \frac{|C_A \cap C_B|}{|C_A \cup C_B|} \quad (1)$$

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 $dist(C_A, C_B)$ を以下のように定義し，距離行列の作成に用いる．

$$dist(C_A, C_B) = 1 - sim(C_A, C_B) \quad (2)$$

5.3 評価実験の概要

実際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．Art Of Illusion[2] のバージョン 1.0 を利用した．このソフトウェアは物体を描画しジオラマを作成するソフトウェアである．構成するソフトウェア部品数 (ファイル数) は 190 であり，利用先部品の類似度と利用元部品の類似度でクラスター分析を行い，機能的に似ている部品や同じような部品から利用されている部品をまとめられるのかを確認する．

5.4 評価項目

Q1 利用先部品の一致度で分類を行った場合，部品群がどのような部品で構成されているのか

Q2 利用元部品の一致度で分類を行った場合，部品群がどのような部品で構成されているのか
それぞれの分類方法によって得られた部品群が，機能的に似ている部品や同じような部品から利用されている部品をまとめることができるのかを確認する．

5.5 適用結果

利用先部品の類似度で分類を行った結果，図 5 の樹形図が得られた．赤線の下で結合した，強く関連していると想定される 13 の部品群を抽出し，中身を確認したところ，大きく分けて 2 つのグループに分けることができた．

- 機能が一致していて類似性が高いグループ
  11 グループがこの群に属していた．グループの中には以下のような機能を持つものが存在した．
  - keyframe に関係する機能を持つ部品の集合．
  - TriangleMesh に関係する機能を持つ部品の集合．クラスター分析の調査対象の幅を広げるとさらに他の同じような機能を持つ部品グループと結合した．
- 機能は似ていないが利用先は類似性が高いグループ
  2 グループがこの群に属する．その内容は以下のようなものであった．
  - ワイヤーフレームを構成する機能を持つものとそのワイヤー上で利用する点を設定するものの組み合わせとなっている．それぞれの部品が持つ機能をお互いが利用する関係になっており，関連性も高い．
  - リストからの反応を返す機能と物体の表示に関係する機能の組み合わせとなっている．直接的な関係はないが Tree を両方とも使用しており利用先部品も共通性がある．

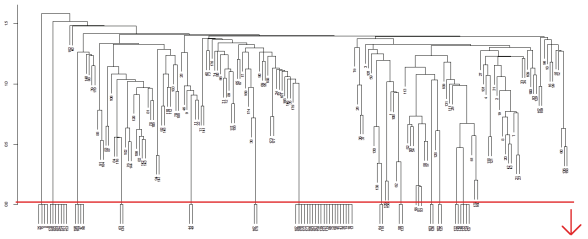

図 5 利用先部品の階層的クラスター分析結果

利用元部品の類似度で分類すると図 6 のような結果が得られた．赤線の下で結合した，強く関連していると想定される 17 グループを抽出し，内部の機能の類似性でさらに分類したところ，大きく 3 グループに分けることができた．

- 機能が一致しているグループ
  11 グループがこの群に属していた．その中には以下のような機能を持つものが存在した．
  - TriangleMesh で作られた物体に対して編集をする機能を実現する部品の集合．TriMeshEditor が利用されている．
  - 物体の配置に関係する機能を実現する部品の集合．Create 系の部品や LayoutWindow などの物体を作成する機能から利用されている．
  - 3 D の線形マッピングを記述する機能でまとめられた部品の集合．LinearMapping3D から利用されている．
- 複数の異なる機能をもつ部品群に分類することができたグループ
  3 グループがこの群に属していた．それらは以下のような機能をもつものでまとめられていた．
  - 物体の構成をするものとエディットツールを表示するものとダイアログボックスを実装するという 3 つの機能で分類することできた．LayoutWindow から利用されている機能で，機能群を構成していると考えられる．
  - グループ内でファイルの読み込みを行うものとウィンドウの表示に関係する機能を持つものでわけることができた．機能の違いのほかに利用元に ModellingApp を持っているかどうかでも分かれている．
  - keyframe に関係する部品と，物体の軸を定める機能を持つ部品で分類することができた．それらは両方とも animation.score と animation.TrackGraph に利用されている．
- 機能の類似性がないグループ

3 グループがこの群に属しており，グループに入った部品が実現する機能と同じ機能を持つ部品がグループになく，他の部品ともあまり関連がない．利用元部品を調べても共通性を見出すのが難しい．

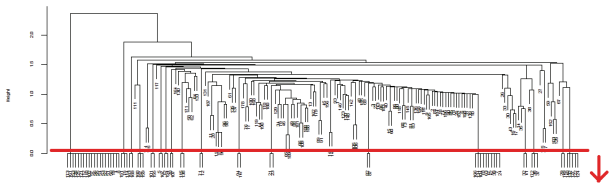

図 6 利用元部品の階層的クラスター分析結果

5.6 評価実験の考察

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，利用元部品のリストを作成し，それらの類似度を元にクラスター分析を行った．利用先部品の共通性で分類したとき，ほとんどの部品群が同じような機能を持つものでまとめられていることを確認した．同じような機能を持つ部品は名称も似ているという特徴もあり，その点からも類似性が高いことがわかる．

利用元部品の類似度を元に分類した場合は同じような機能を持つグループに分かれる場合が多いが，共通性を見出すのに利用元の部品の情報が必要な場合や利用元を見ても共通性を見いだせない場合があった．部品同士の機能の関連のみに注目するのではなく，機能群として捉えることや共通の利用元部品の役割が重要となることが予想される．

適用事例を増やし，手法によって得られる知見の一般性を高めることが求められる．また，利用先部品の集合，利用元部品の集合，それぞれ単独の場合の事例のみであるが，複数の集合を組み合わせた場合の結果も考察したい．

6. まとめ

本研究では，利用関係やコピーされた部品などの部品間の関係に着目し，それらの関係に基づいてソフトウェア内で同じような性質を持つ部品を部品群に分類することを目的とした二通りの分類手法を提案した．部品グラフ上で各部品の入出力辺の数などを用いて値の似ている部品ごとに分類する方法，利用先（利用元）の部品がどれだけ一致しているかという類似度から各部品間の距離を求めて分類する方法，それぞれに対して実際のプロジェクトに対して適用した結果を示し，それぞれの手法によって，ソフトウェアのファイルがどのように分類されるかを紹介し，似たような機能を実現する部品，ある部品で実現している機能群に含まれる部品という形で，ソフトウェアを構成する部品を部品群として抽出することができることを確認した．

現状は，それぞれの手法で 1 プロジェクトにおいての分析を行ったのみであるので，分析対象プロジェクトを増やしたうえでそれぞれの手法の一般性を確認していくとともに，今後，それぞれの結果の比較や，得られた部品群の特性の違いなどを考察していきたいと考えている．

謝辞 本研究は，2016 年度南山大学パッヘ研究奨励金 I-A-2 の研究助成を受けている．この論文は，今後発表予定の卒業研究 [13], [14] の内容をまとめたもので，濱島直輝君，日下光君，堀貴行君，望月俊希君，後藤彗君に感謝の意を表します．筆者は，これらの卒業論文の指導教員としてテーマの設定，研究指導などを全面的に行っています．

参考文献

[1] Antoniol, G., Villano, U., Merio, E. and Penta, M. D.: Analyzing cloning evolution in the Linux kernel, *Information and Software Technology*, Vol. 44, No. 13, pp. 755–765 (2002).
[2] Art Of Illusion: `http://sourceforge.net/projects/aoi/`.
[3] Classycle: `http://classycle.sourceforge.net/`.
[4] Constantinou, E., Kakarontzas, G. and Stamelos, I.: Towards Open Source Software System Architecture Recovery Using Design Metrics, *Proceedings of the 15th Panhellenic Conference on Informatics*, pp. 166–170 (2011).
[5] Kamiya, T., Kusumoto, S. and Inoue, K.: CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code, *IEEE Transactions. Software Engineering*, Vol. 28, No. 7, pp. 654–670 (2002).
[6] Krueger, C.: Software Reuse, *ACM Computing Surveys*, Vol. 24, No. 2, pp. 131–183 (1992).
[7] Mondal, M., Roy, C., Rahman, M., Saha, R., Krinke, J. and Schneider, K.: Comparative Stability of Cloned and Non-cloned Code: An Empirical Study, *Proceedings of the 27th ACM Symposium on Applied Computing*, pp. 1227–1234 (2012).
[8] Turtle Sport: `http://turtlesport.sourceforge.net/`.
[9] Yoshida, N., Higo, Y., Kamiya, T., Kusumoto, S. and Inoue, K.: On Refactoring Support Based on Code Clone Dependency Relation, *Proceedings of the 11th IEEE International Software Metrics Symposium*, pp. 16:1–16:10 (2005).
[10] Zhangs, Q., Qiu, D., Tian, Q. and Sun, L.: Object-oriented software architecture recovery using a new hybrid clustering algorithm, *Proceedings of the Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 2546–2550 (2010).
[11] 小堀一雄，山本哲男，松下 誠，井上克郎：コードの静的特性を利用した Java ソフトウェア部品類似判定手法，電子情報通信学会論文誌 D， Vol. J90-D(4), pp. 1158–1160 (2007).
[12] 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).
[13] 堀貴行，望月俊希， 後藤彗：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，南山大学情報理工学部 2016 年度卒業論文 (2017 発表予定).
[14] 濱島直輝， 日下光：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，南山大学情報理工学部 2016 年度卒業論文 (2017 発表予定).