利用部品の共通性に基づくソフトウェア分類手法
類似度計算手法に関する考察

2016SE008 藤田翔大　2016SE071 清水太智　2016SE084 戸本了太
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．我々の研究グループでは，利用先の一致度に基づいて部品間の距離行列を作成し，階層的クラスタリングを行うことで機能的に類似した部品を類似部品群として抽出する手法を提案した [4]．評価実験からは，手法の有効性は確認できたが精度向上の余地があることも確認できた [5].

本研究では，精度向上のための手法として距離行列の距離の計算方法の改良を目的とした研究を行う．改良のアプローチとして，類似度計算を利用部品の一致割合でなく，一致利用部品数に基づいて定義し，距離行列を作成することで，結果がどう変化するか実験を行い，評価する．さらに，利用関係の局所性を考慮した場合に結果がどう変化するか実験を行い，評価する．「できるだけ多くの部品を対象に，実現している機能面で類似性を持つ部品をグループ化する分類手法」を確立することで，開発者がよりソフトウェアを効率よく理解することを支援できると考える．

2　背景技術

2.1　ソフトウェア部品と部品グラフ

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のモジュールの一部を指す [1]．各クラスのソースコードを記述しているファイルを部品とみなし，各部品を構成要素とする部品グラフを構築する．部品グラフ上の頂点は各部品を表し，辺は部品間の関係として利用関係を表現する．ある部品 A が他の部品 B を利用している場合，A から B への利用関係が存在しているとみなし，頂点 A から B への有向辺で表現する．

2.2　関連研究

ソースコードからパターンを抽出し，ソフトウェア理解支援に活用する研究が行われている．Zhong らは [2] で，ソースコードから API の利用順を抽出し，API の利用方法の学習に活用するシステムを提案した．また，Li らは [3] で，C の中から関数呼び出しの実行順を取り出して，プログラミングのルールとしてルールから逸脱している記述があるかを調べる仕組みを提案している．

我々の研究グループではソフトウェア部品の利用先がどれだけ一致しているかから各部品の類似度を計算し，距離行列を作成し階層的クラスタリングを行い得られた樹形図を基にソフトウェア部品を分類する手法を提案した [4]．評価実験として，得られた部品群中の部品がどれだけ関連していたか，本来含むべき部品をどれだけ含むことができたかという適合率や再現率の観点から評価を行い，有効性を確認した．橋本らは，部品群中の部品の役割の共通点と部品群中の部品が共通して利用している部品の役割が多くの部品で一致していること，多くの部品から利用されている部品について，その部品を利用している部品の集合が適切に分類されていることを示した [5]．最適であると思われる分類結果のうち一部の部品は含んでいないという点で改善の余地が存在するが，全体としてソフトウェア部品を適切に分類できていることを示した．

3　ソフトウェア内の部品の利用関係に基づく分類手法における類似度計算手法

3.1　研究の動機と改善のアプローチ

[4] の手法では, ソフトウェア部品の利用先がどれだけ一致しているかから各部品の類似度を計算し，以下の手順で距離行列を作成し評価を行っている．

1. 分析対象のソフトウェアを Classycle で分析して，各部品の利用関係を入手する．
2. 各部品について，ソフトウェア内で定義された部品の利用関係を入手し，利用部品の集合を作成する．
3. 各部品ごとに 2. で求めた利用部品集合の類似度を求め，距離を計算し，各部品間の距離をもとに距離行列を作成する．
4. 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．得られた樹形図において，まとまりになっている部品から類似部品群を抽出する．
5. 分類した結果の意味づけを行い関連性を調査する．

3. の距離計算では，2 つの部品の類似度を求める際に，共通して利用している部品の積集合の部品数を 2 つの部品が利用している部品の和集合の部品数で割っている．この計算手法では大きな部品と小さな部品の間の類似度は，和集合中の部品数と積集合中の部品数から常に低い値となる．類似した部品を検出する際に和集合が不必要である可能性がある．一致利用部品数のみで類似度を定義しなおすことで，精度の向上が見られるかを調査する．

また，[4] の手法では，利用関係について局所性は一切考慮されていない．そこで，「利用する部品の数が少ない部品への利用関係は局所性を持つ」と仮説を置き，再定義した手法上で局所性を反映させたときに，分類結果がどう変

わるかを調査し，局所性を考慮することの必要性を確認する．精度向上を目的とした手法を適用し，分類手法としてより適切な手法とすることを目的とする．

3.2 部品間の類似度と距離の定義の変更について

2 つの部品間の類似度とそれに基づく距離の計算手法を次のように定義しており，以下「旧方式」と呼ぶ．分析対象のソフトウェア部品の集合を C とする．ある部品 $A(\in C)$ が利用している部品の集合を O_A，ある部品 $B(\in C)$ の利用している部品の集合を O_B とする．

$$sim(A,\ B) = \frac{|O_A \cap O_B|}{|O_A \cup O_B|}$$

$$dist(A,\ B) = 1 - sim(A, B)$$

提案手法では，一致利用部品数 (com) と部品間の相違度 (diff) を定義する．C 中のすべての部品の組み合わせにおける $com(X \in C,\ Y \in C)$ の最大値を $MAX(X,\ Y)$ とする．$1 \geq \mathrm{diff}(A,\ B) \geq 0$ となり値域自体は変わらない．この計算手法を「新方式」と呼ぶ．新方式では，部品間の相違度 (diff) を距離行列中の距離とみなして距離行列を作成する．

$$\mathrm{com}(A,\ B) = |O_A \cap O_B|$$

$$\mathrm{diff}(A,\ B) = 1 - \frac{|\mathrm{com}(A,\ B)|}{MAX(X,\ Y) + 1}$$

ただし，$\mathrm{diff}(A,\ A) = \mathrm{diff}(B,\ B) = 0$

3.3 局所性を考慮した類似度計算手法について

局所性を考慮した類似度計算手法を「利用する部品の数が少ない部品への利用関係は局所性を持つ」と仮説を置いた上で定義する．局所性を考慮した手法は，新方式をベースにしており，一致利用部品数 (com) の計算方法を以下のように変更する．変更した com によって得られる値を一致部品利用度と定義する．

- A,B が共通して利用している部品それぞれについて，「その部品が (A,B を含めて) いくつの部品から利用されているか」を求める．
- その部品が N 個以下の部品から利用されている場合に，その利用関係は局所性を持つと判断する．
- 局所性をもつ 利用関係の数 $*\ (X - 1)$ の値を com(A, B) に追加する．

この方法の場合，局所性のない利用関係は重み 1，局所性のある利用関係は重み X として，com の値が計算される．diff の計算方法は同一とし，距離行列を作成する．この方法による分類を「局所性を考慮した分類手法」と呼ぶ．それぞれの方法で得られた距離行列に対し階層的クラスタリングによって得られた樹形図を比較し評価する．

4 評価実験

4.1 分析対象アプリケーション

本研究では，JavaPlot と barbecue という 2 つの Java アプリケーションに対してそれぞれ適用実験を行った．JavaPlot は Java プログラム上でグラフを生成するためのライブラリで，内部ではグラフを作成するソフトウェアである GNUPlot を実行してグラフを作成しており，51 のソースファイル（部品）で構成されている．barbecue は国際規格に適合したバーコードを作成する Java アプリケーションで，59 のソースファイルで構成されている．要旨集では JavaPlot に対して適用した結果を紹介する．

4.2 比較の方法

実験では，JavaPlot と barbecue それぞれに対して旧方式，新方式の計算手法を適用し，得られた部品群を示す．部品群を得る際には部品群内のすべての部品や関連のある部品となるように部品を追加する．得られた最大の集合をそれぞれ部品群とみなしている．部品の中身を調査したうえで得られた樹形図上での部品群を示したのちに新方式と旧方式で得られた部品群同士を比較する．それぞれの樹形図上で得られた部品群をもう片方の樹形図上に配置することで，どちらの分類がより適切であったかを調査する．局所性を考慮した場合とそうでない場合についても同様の方法で評価を行い，局所性を考慮することでどのような変化がおこりうるかについて考察する．

4.3 JavaPlot に対する旧方式と新方式での分類結果の比較

JavaPlot に対して旧方式を適用したときの樹形図を図 1 に示す．旧方式の結果として，JavaPlot に関して 10 個の部品群が形成され，総部品数 51 個のうち 32 個が部品群に分類され，のこり 19 個が分析対象から外れていた．

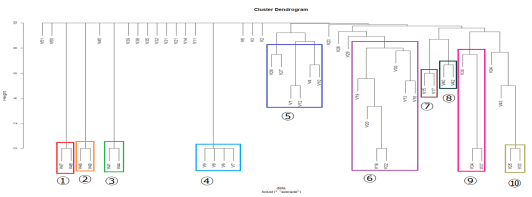

図 1 JavaPlot に対して旧方式を用いて分類した樹形図と得られた類似部品の集合

表 1 JavaPlot 旧方式を用いて得た部品群の紹介

部品群番号	部品数	主な部品	共通利用部品（部品数）	部品の役割
1	2	PostecriptTerminal，SVGTTerminal	TextFileTernimal(1)	出力のフォーマット
2	2	ImageTerminal，TextFileTerminal	FileTerminal(1)	ファイル形式の定義
3	2	CustomTerminal，FileTerminal	ExpandableTerminal(1)	端末の定義
4	4	LongDataParser，DoubleDataParser	NumericDataParser(1)	データパーサー
5	6	DataSetPlot.ArrayDataSet.GenericDataSet	× (0)	データセット
6	7	GrapLayout，AutoGraphLayout，GNUPlot	× (0)	グラフ出力に関連
7	2	GNUPlotExec，JavaPlot	GNUPlotException(1)	Gnuplot に関連
8	2	Jplot，DafaultTerminal	GNUPlotTerminal (1)	端末
9	3	NamePlotColor，RgbPlotColor	PlotColor(1)	グラフの色
10	2	Axis，FillStyle	PropertiesHolder(1)	グラフに関連

JavaPlot に対して新方式を用いて分類したときの樹形図を図 2 に示す．新方式では，JavaPlot に関して 9 個の部品群が形成され，51 個の部品のうち 35 個が部品群に分類され，のこり 16 個が分析対象から外れていた．

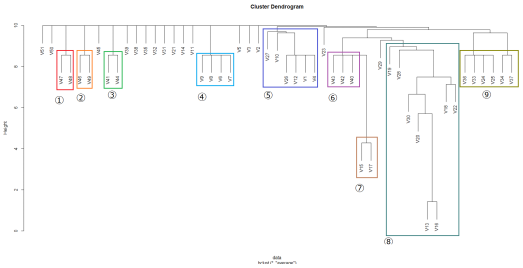

図 2　JavaPlot に対して新方式を用いて分類した樹形図と得られた類似部品の集合

表 2　JavaPlot 新方式を用いて得た部品群の紹介

部品群番号	部品数	主な部品	共通利用部品（部品数）	部品の役割
1	2	PostecriptTerminal，SVGTTerminal	TextFileTerminal(1)	出力のフォーマット
2	2	ImageTerminal，TextFileTerminal	FileTerminal(1)	ファイル形式の定義
3	2	CustomTerminal，FileTerminal	ExpandableTerminal(1)	端末の定義
4	4	LongDataParser，DoubleDataParser	NumericDataParser(1)	データパーサー
5	6	DataSetPlot.ArrayDataSet.GenericDataSet	× (0)	データセット
6	3	DefaultTerminal，ExpandableTerminal	GNUPlotTerminal(1)	端末
7	2	GNUPlotExec，JavaPlot	GNUPlotException(1)	Gnuplot に関連
8	8	GNUplot，Page，AutoGraphLayout	× (0)	グラフ出力に関連
9	6	Axis，AbstractPlot，RgbPlotColor，PlotStyle	× (0)	グラフ描写に関連

二つの方法で得られた部品群の差を確認するために，旧方式で得られた類似部品群を新方式の樹形図上に配置した図を図 3 に示したところ，次のような点が確認できた．

- 部品群 1〜5，7 は結合する点は樹形図上で上部になったが，同一の部品からなる独立した部品群として得られている．
- 部品群 6 は 1 つだけ部品が分離した．分離した部品は GraphLayout で，Graph という部品を挟んで分離した形になっている．Graph 自身もその部品群に含まれるべき部品であったので，旧方式では，本来含まれるべきだった部品を含めていない．
- 部品群 8 は JPlot と DefaultTerminal で構成されているが，新方式の樹形図ではもう 1 つの類似部品 ExpandableTerminal が存在することがわかる．旧方式では，本来含まれるべきだった部品を含めていない．
- 部品群 9 は NamedPlotColor や RgbPlotColor など，部品群 10 は Axis と FileStyle で構成されている．これらは JavaPlot 内のグラフに関係している点において互いに類似しており，かつ AbstractPlot を含んだ形で図 2 の部品群 9 のように 1 つの部品群とみなされるべきであった．旧方式では，本来含まれるべきだった部品を含めていないことが分かった．

上記の 3 点で含まれる部品の観点から違いが見られた．それぞれ対応する部品群を前述のような形で比較したが，いずれの場合も新方式による分類結果の方が良く，この事例

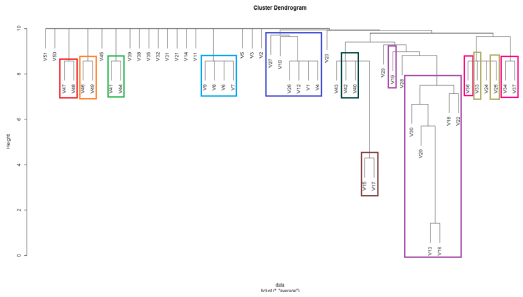

図 3　旧方式で得られた類似部品群を新方式で得られた樹形図上に配置した図

では新方式のほうが適切であった．

4.4　JavaPlot に対する新方式と局所性を考慮した方式での比較の結果

JavaPlot に対して局所性を考慮した手法を用いて分類したときの樹形図を図 5 に示す．新方式 (図 2) では，JavaPlot に関して 9 個の部品群が形成され，51 個の部品のうち 35 個が部品群に分類された．局所性を考慮した手法では，10 個の部品群が形成され，35 個が部品群に分類された．

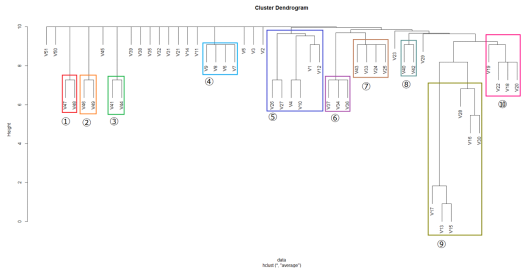

図 4　JavaPlot に対して局所性を考慮した手法を用いて分類した樹形図と得られた類似部品の集合

表 3　JavaPlot 局所性を考慮した手法を用いて得た部品群の紹介

部品群番号	部品数	主な部品	共通利用部品（部品数）	部品の役割
1	2	PostecriptTerminal，SVGTTerminal	TextFileTerminal(1)	出力のフォーマット
2	2	ImageTerminal，TextFileTerminal	FileTerminal(1)	ファイル形式の定義
3	2	CustomTerminal，FileTerminal	ExpandableTerminal(1)	利用部品の環境を利用
4	4	LongDataParser，DoubleDataParser	NumericDataParser (1)	データパーサー
5	6	DataSetPlot.ArrayDataSet.GenericDataSet	× (0)	データセット
6	3	RgbPlotColor，NamedPlotColor，PlotStyle	PlotColor(1)	グラフの色
7	4	FillStyle.AbstractPlot，Expandableterminal	PropertiesHolder(1)	グラフ作成
8	2	Jplot，DafaultTerminal	GNUPlotTerminal (1)	端末に関連
9	6	JavaPlot.GNUPlot，GNUPlotExec.Page	× (0)	中枢部品
10	4	GraphLayout，AutoGraphLayout	Page(1)	グラフの配置方法

二つの方法で得られた部品群の差を確認するために，新方式で得られた類似部品群を局所性を考慮した分類手法で得られた樹形図上に配置した図を図 6 に示したところ，次のような点が確認できた．

- 部品群 1〜5 は構成は変わらずに独立した部品群とし

て得られている．

- 部品群 6 から 1 つだけ部品が分離した．分離した部品は ExpandableTerminal で局所性を考慮した場合では新方式の部品群 9 の FillStyle などと様々なパラメータを利用してグラフを作成している点で同じ部品群として結合した．ExpandableTerminal は新方式では GNUPlot に関係している大まかな類似点によりまとまっていたので，局所性を考慮した分類手法がより細かい役割の類似点でまとまっていることがわかる．
- 部品群 8 から 1 つだけ部品が分離した．分離した部品は GNUPlot で局所性を考慮した場合では新方式の部品群 7 の JavaPlot などで，JavaPlot の中枢の機能を持つという類似点で同じ部品群として結合した．新方式では，グラフの出力に関わるという大まかな点でまとまっていたので，局所性を考慮した分類手法がより細かい役割の類似点でまとまっていることがわかる．

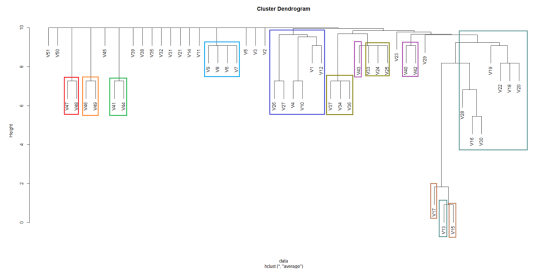

図 5　新方式で得られた部品群を局所性を考慮した手法で得られた樹形図上に配置した図

上記の 2 点において違いが見られた．それぞれ対応する部品群を比較したが，いずれの場合も新方式での分類結果では大まかにまとめられていた部品群が、局所性を考慮した手法による分類結果ではより具体的な役割ごとに細かく分類されていた．

5　考察

5.1　類似部品数による分類について

新方式と旧方式を比較した場合，結合点が樹形図上で上になることはあっても，得られる部品群の大多数は，構成する部品もほぼ一致し，同じ集合が得られた．組み合わせが変わる場合というのは，ある特定の部品Aを利用している部品の集合をさらに詳しく分けたときのように，分類結果をさらに詳しく分けるときに生じる．旧方式の場合は，類似度計算で利用部品数が多い大きい部品と利用部品数が少ない小さい部品は必ず分かれるので目的に反した分類を行っている可能性がある．

5.2　局所性を考慮した分類手法について

局所性を考慮しなかった場合，大まかにみて機能面で類似している部品同士がくっついていることが多かったが，局所性を考慮した分類手法では，より細かい役割の類似点でまとまるような傾向が見られた．局所性を持った部品の利用でまとまることでその部品を利用しているという共通点が得られているので，分類結果については類似点の根拠を示しやすくなると考えられる．今後の課題として，局所性の判定がこの方法で十分かを，いくつかの代替案での結果と比較して評価する必要がある．tf-idf 形式のような局所性の重みづけ方法についても導入の検討が必要である．

6　まとめ

本研究では，距離行列の計算の方法を定義しなおした場合と，さらに局所性を考慮した場合とで，クラスタリングの結果として得られた部品群がどのように変化するかを，樹形図上で示すことで比較を行い，評価した．得られる部品群の質が良くなる場合もあれば，結果的に同じであった場合もあった．局所性を考慮した分類手法の場合，考慮しなかった場合と比べてより細かく，具体的に何を使っているかで分類することができており，類似点の根拠を示しやすくなっていた．今後の課題として，他のソフトウェアでも同様の評価実験を行い，今回得られた傾向が一般性を持つかについての検証を行うとともに，手法の精度を向上させるための方法について考察する必要がある．

参考文献

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

[2] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei, ”Mapo: Mining and recommending api usage patterns,”in proceedings of the 23rd European Conference on Object-Oriented Programming (ECOOP 2009), 2009, pp. 318-343.

[3] Z. Li and Y. Zhou, “Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code,” in proceedings of the 10th European software engineering conference,2005, pp. 306-315.

[4] Reishi Yokomori, Norihiro Yoshida, Masami Noro, Katsuro Inoue: ”Use-Relationship Based Classification for Software Components”, Proceedings of the 6th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2018), pp.59-66, 2018.

[5] 橋本敬太，川瀬史也 ：“利用部品の共通性に基づくソフトウェア部品分類手法の-共通して利用している部品の観点から評価-”，南山大学情報理工学部 2018 年度卒業論文，2019.