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評価
—共通して利用している部品の観点からの評価—

2015SE010　橋本敬太　　2015SE041 川瀬史也
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，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．過去に行われた研究では，類似度を元にソフトウェア部品を分類することで似たような部品を抽出する手法 [1] に対し，本来含まれるべき部品のうちどれだけを分類された部品群において含んでいたかという観点から適合率，再現率の評価を行った．その結果，利用先の部品の一致度に基づいて分類した場合，適合率，再現率ともに高い割合を示すことを確認した [2]．しかし，[1] の手法をどうソフトウェア理解に生かすかの議論を行っておらず，その観点からの評価が必要である．

本研究では，「得られた部品群内の部品が共通して利用している部品を理解の軸として活用する」アプローチが有効であることを示すことで，提案手法をソフトウェア理解に活用できることを確認する．具体的には，部品群中の部品が実現する機能についての共通性が，共通で利用している部品が提供する機能と一致しているかを調査する．また，多くの部品から利用されている部品について，それらがどう分類されるべきかを調べた上で，その集合が樹形図にどう表れるかを調査する．これらの調査を通じて，ソフトウェアの構成を理解しようとしたときに，提案手法により得られた部品群が示す類似性を活用することで，効果的に理解することを支援できることを示す．

2　背景技術

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上で，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 [3]．本研究では 1 つの Java ソフトウェア全体を分析対象とし，それぞれの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．本研究ではある部品 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，部品グラフ上で頂点 A から頂点 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 [4] で，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 [5] で，C の中から関数呼び出しの実行順を取り出して，それを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どうかを調べる仕組みの研究を行っている．

ある部品においてある機能を実現しようとする場合，ほかの部品で提供されている機能も利用しながら目的となる機能を実現する．我々は，ソースコードからクラス間の利用関係を抽出し，ソフトウェア理解支援に役立てる方法を研究している．堀らは [1] で，利用先に関して 2 つの部品が利用している部品が一致している割合が高いほどそれらの部品同士は目的や役割が似ている部品となるのではないかと考えた．[1] では，ソフトウェアの利用先がどれだけ一致しているかから各部品対の類似度を計算し，類似度をもとにソフトウェアを以下の手順で分類することで，機能や役割が似ていると思われる部品を抽出する手法を提案した．

1. 分析対象のソフトウェアを分析して，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，各部品毎に利用先部品の集合を作成する．
3. 各部品対ごとに類似度を求め，距離を計算する．
4. 各部品間の距離関係をもとに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．得られた樹形図において，まとまりになってる部品から類似部品群を抽出する．
6. それぞれの類似部品群内の部品を調査し，どのような点で類似しているかを調査する．

間瀬ら [2] は，[1] の手法での分類結果が類似した部品をどれだけ部品群内に含むことができたかを調査するために，適合率，再現率の観点から評価を行った．結果として，利用先の一致度を用いて分類した場合，適合率，再現率ともに高い割合を示すことができ，分類結果として含まれるべき部品の多くを含むことができていたこと確認した．

3　ソフトウェア内の部品の利用関係に基づく分類手法

3.1　現在の課題

間瀬ら [2] の研究では，ソフトウェアの利用先の一致度に基づいて分類を行う手法において適合率，再現率の観点からの評価を行い，分類結果として含まれるべき部品を部品群に多く含んでいることを確認した．しかし，提案手法をどのようにソフトウェア理解に生かすかの議論を行っておらず，その観点からの評価が必要である．本研究では，

「得られた部品群内の部品が共通して利用している部品を理解の軸として活用する」アプローチが有効であることを示すことで，提案手法をソフトウェア理解の支援に活用する方法として利用できることを示す．また，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提案手法によって適切に分類できていることを示すことで，ソフトウェア内の部品を適切に分類できていることを [2] とは異なる観点から示す．ソフトウェア部品の動作を理解するために類似した部品を提示することで，似た動作の部品の存在を事前に把握でき，ソフトウェアのサブシステムの動作や全体の動作を理解する作業を支援できる．

3.2 評価項目

以下のリサーチクエスチョンを設定し評価実験を行う．

1. 分類結果として得られた部品群において，共通して利用している部品がどう関係しているか？

部品群それぞれにおいて，部品群中の部品の共通性が，共通で利用している部品の役割と一致しているかを調査する．部品群中の部品の共通性が，共通して利用している部品の役割と一致していれば，部品群の役割やまとまりを効率よく把握することができ，ソフトウェアの理解の手がかりを効率よく得ることができる．

2. ソフトウェア内で多く利用されている部品について，その部品を利用する部品を提案手法で適切に分類できるか？

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を求める．次に部品の利用方法，各部品が実装しているメソッド，コードクローンの分布状況などから，その部品の集合から抽出されるべき部品群をあらかじめ抽出する．この抽出されるべき部品群を提案手法によって得られた樹形図の中で部品群となっているかを調査し，ソフトウェアの分類結果が適切であることを示す．

4 評価実験

4.1 実験の内容

jlgui と jfm という 2 つの Java アプリケーションに対して適用実験を行った．[2] と同様に，階層的クラスター分析においては，群平均法を用いた．jlgui はイコライザ機能を提供する Java アプリケーションで，70 のソースファイル（部品）で構成されている．jfm はファイル管理をサポートする Java アプリケーションで，85 のソースファイル（部品）で構成されている．

実験 1 では，樹形図上で得られた部品群を示すとともに，得られたそれぞれの部品群が持つ共通点と部品群が共通で利用している部品の関係を示す．それぞれの部品で部品が実現している代表的な機能を抽出し，得られた部品群内の部品が実現している機能について共通点を調査する．その共通点がそれらの部品で共通して利用している部品が提供する機能と一致するかを調査する．実験 2 では，多くの部品から利用されている上位 10 個の部品に対して，その部品を利用している部品を部品の利用方法，各部品が実装しているメソッド，コードクローンの分布状況などの観点から分類し，それらの部品を分類したときに，抽出されるべき部品の集合を求める．その部品集合が樹形図上で部品群となっているかを調査する．

4.2 評価実験 1 の結果

jlgui と jfm に対して手法を適用したときの樹形図をそれぞれ図 1，図 2 に示す．また，表 1，表 2 では，それぞれの部品群の ID，部品群内の部品数，それらが共通して利用している部品の代表例と共通点，部品群としたときの樹形図上での高さを示し，部品の共通点と共通して利用している部品の目的が一致しているかを判定した結果とともに示す．結果として，jlgui に関して 7 個の部品群が形成され，うちの 5 個の部品群で部品群中の部品の共通点が，共通で利用している部品の役割と一致していた．一方で，jfm に関しては 14 個の部品群が形成され，それらの部品群のすべてにおいて，部品の共通点と，共通で利用している部品の役割が一致していた．jlgui では，大きな部品群が得られやすかったが，jfm では細かい集合に分かれやすいという傾向の違いが得られた．全体として，部品群が共通して利用している部品の役割とその部品群内の部品の共通性が多くの場合一致していた．また，提案手法で得られた部品群の形成の仕方はソフトウェアによってまちまちで部品群のサイズの大小も異なることを確認した．

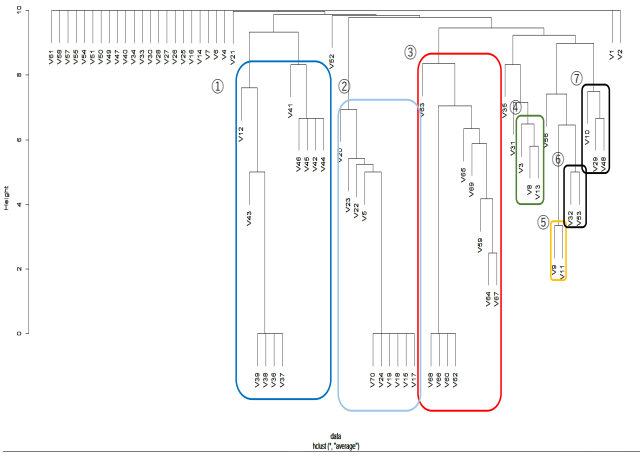

図 1 (jlgui) 利用先の一致度で分類した結果の樹形図

4.3 評価実験 2 の結果

jlgui と jfm のそれぞれについて多くの部品から利用されている部品を選出した．選出した部品を利用している部品について，部品の利用の用途，各部品が実装しているメソッド，コードクローンの分布状況などから，その部品の集合から抽出されるべき部品群をあらかじめ抽出し，それらが樹形図上でまとまって分布しているかどうかを調査した結果の一部を示す．表 3，表 4 は抽出されるべき部品群

表 1　部品群ごとの部品と共通して利用している部品の一覧 (jlgui)

部品群の ID	部品数	共通で利用している部品	部品の共通点	打ち切った高さ	判定
1	11	TagInfoDialog	音楽ファイルの形式	9.4	○
2	10	AbsoluteConstraints	GUI の実現	7.2	○
3	10	PreferenceItem，PlayerUI	設定ファイル	9.0	○
4	3	PlayerUI，Skin など (15)	ユーザーインターフェース	6.8	○
5	2	Playlist ，Config	プレイリスト	4.0	○
6	2	Config	GUI の実現	なし	×
7	2	PlaylistItem，Playlist	音楽ファイルの検索	なし	×

表 2　部品群ごとの部品と共通して利用している部品の一覧 (jfm)

部品群の ID	部品数	共通で利用している部品	部品の共通点	打ち切った高さ	判定
1	6	BroadcastListener	イベント処理	7.2	○
2	8	BroadcastEvent	イベント処理	6.0	○
3	2	Options	設定管理	6.0	○
4	3	ConfigurationEventsQueue など (4)	環境設定	6.0	○
5	2	ConfigurationDialog など (5)	環境設定	7.2	○
6	4	FSException ，JFMFile	ファイルシステム	7.2	○
7	6	JFMFile	ダイアログ表示	6.0	○
8	5	JFMFile ，Options	ファイル管理	7.2	○
9	4	Broadcaster など (6)	ユーザー操作	5.0	○
10	3	FileViewDialog など (6)	ファイルの編集	5.5	○
11	2	JFMView など (7)	ファイルの一覧を見る	4.0	○
12	3	ChangeDirectoryEvent など (3)	ファイルの処理	4.8	○
13	2	CopyAction など (7)	設定管理	8.0	○
14	2	JFMFileSystemt など (16)	ファイル編集	2.0	○

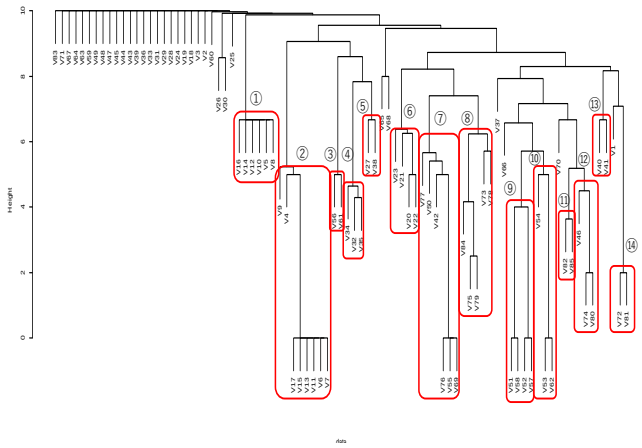

図 2　(jfm) 利用先の一致度で分類した結果の樹形図

を示したもので，それぞれの分別された類似部品の分別結果を表し，ファイル名，対象のファイルを利用している部品数，抽出した部品群，抽出した部品群の部品数，抽出した理由，代表のクラス，樹形図上でまとまって分布していたかどうかの判定結果を示す．判定結果について，樹形図上ですべてまとまっているものに対しては◎，6 割以上，もしくはまとまりとして考えられるものに対して○，6 割未満，もしくは一部まとまっているものに対して△，全くまとまっていないものに対して×として判定する．jlgui では，抽出した部品群の多くでまとまった分布をみせ，部品群としてのまとまりが形成されていた．一方 jfm では，抽出した部品群のほとんどでまとまった分布をみせ部品群を形成していたが，一部部品群において複数のクラスターに分かれて分布していたことが確認できた．抽出した部品群のほとんどが樹形図上でまとまった分布を見せた．提案手法の精度の向上の余地はあるが，全体として適切な分類はできていると考えられる．

5　考察

5.1　評価実験 1

利用先の一致度を，共通して利用している部品を用いて部品群中の部品の共通性と共通で利用している部品の役割と一致しているかで判定した結果は，jlgui，jfm ともに比較的一致してるといえる結果になった．jlgui では，大きな範囲をとった部品群が形成され共通して利用している部品の数が少ない傾向にあった．一方 jfm では，大きな範囲をとった部品群はあまり形成されず，細かく分かれて部品群が形成される傾向が見られ，低い閾値での部品群が多く，共通して利用している部品の数が多い傾向にあった．部品群内で共通して利用している部品の数が多いほど部品群の役割が理解しやすいが，部品群内の部品数が増えるほど共通して利用している部品の数が少なくなり，共通して利用している部品の数を多くすると小さな部品群となることが確認できた．このことから「得られた部品群内の部品が共通して利用している部品を理解の軸として活用する」というアプローチが有効であると言えるが，どのように部品群を機械的に設定するかについてはさらなる実験が必要である．

5.2　評価実験 2

jlgui では，抽出した部品群の多くが樹形図上でまとまった分布をみせ部品群としてのまとまりが形成されていた．jfm では，似た部品名やメソッドを持つ部品はまとまった分布傾向を見せたが，一部の同じ部品から派生している部品に関してまとまった分布がみられない結果となった．利用の用途によっていくつも抽出すべき部品群が存在するこ

表 3 多く利用されている部品を利用している部品の分布 (jlgui)

ファイル名	利用部品数	部品群	部品群の部品数	抽出理由	クラス代表例	結果
AbsoluteConstraint	15	1-1	3	同じ部品から派生，同じインターフェースを実装	EqualizerUI	◎
		1-2	7	メソッドが似ている	ActiveJBar	○
Config	15	2-1	3	同じ部品から派生，同じインターフェースを実装	EqualizerUI	◎
		2-2	2	同じ部品から派生	StandalonePlayer	×
		2-3	3	同じインターフェースを実装	SkinPreference	○
PlayerUI	10	3-1	2	同じインターフェースを実装	EqualizerUI	◎
		3-2	3	同じ部品から派生	TagSearch	×
		3-3	4	同じ部品から派生，同じインターフェースを実装	DevicePreference	○
TagInfo	10	4-1	4	メソッドが似ている	APEInfo	◎
		4-2	2	同じインターフェースを実装	PlayerUI	◎
PreferenceItem	9	5-1	8	同じ部品から派生	EmptyPreference	○

表 4 多く利用されている部品を利用している部品の分布 (JFM)

ファイル名	利用部品数	部品群	部品群の部品数	抽出理由	クラス代表例	結果
JFMFile	33	1-1	3	メソッドが似ている，同じ部品から派生	JFMFileSystem	◎
		1-2	7	メソッドが似ている	CopyAction	◎
		1-3	2	メソッドが似ている，同じ部品から派生	FileViewPanel	×
		1-4	4	メソッドが似ている，同じ部品から派生	BriefView	△
		1-5	3	メソッドが似ている，同じ部品から派生	BriefViewListRenderer	○
Option	25	2-1	3	メソッドが似ている，同じ部品から派生	ColorConfigurationPanel	◎
		2-2	9	メソッドが似ている	CopyAction	◎
		2-3	2	メソッドが似ている，同じ部品から派生	FileViewPanel	×
		2-4	2	メソッドが似ている，同じ部品から派生	BriefViewListRenderer	◎
		2-5	3	メソッドが似ている，同じ部品から派生	BriefView	○
Broadcaster	17	3-1	4	メソッドが似ている	CopyAction	◎
		3-2	2	メソッドが似ている，同じ部品から派生	FileViewPanel	×
		3-3	4	メソッドが似ている，同じ部品から派生	BriefView	△

とを確認し，提案手法を用いると抽出すべき部品の多くは検出できていたと考えられる．ただし，抽出すべき部品群の中のいくつかは正しく分類できておらず，原因の究明と精度の向上についての情報が必要である．

5.3 提案手法を用いたソフトウェア理解

本研究の手法を用いたソフトウェアの作成を考えると，ソフトウェアのパッケージ階層から閲覧対象のコードとそれに付随して似たコードの一覧をリストアップするようなコードブラウザが良いのではないかと考えられる．次に見るべきコードをリストアップする際，コードクローンのみならず，継承や実装，利用や被利用の関係といった条件を考慮することにより類似した部品をリストアップすることで開発者が次に必要になりそうなコードを入手しやすくなり，類似した部品をリストアップする際の精度の向上か同じような機能を実現しているソースコードの場所を事前に理解することが出来るようになり，対象のソフトウェアに初めて触れるプログラマーでも効率的にソフトウェアを理解できるようになるのではないかと推測する．

6 まとめ

本研究では，分類結果として得られた部品群において，部品群中の部品の共通点が共通して利用している部品の役割と一致しているかの調査と，多くの部品から利用されている部品についてそれを利用している部品がどう分類されるべきかを調べた上で，その集合が樹形図にどう表れるかの調査を行った．評価実験の結果から，分類結果として得られた部品群のほとんどにおいて部品群中の部品の共通点と共通で利用している部品の役割が一致している結果となった．また，抽出されるべき部品群の多くで樹形図上でもまとまった分布結果を示し，提案手法を用いたソフトウェアの分類結果が適切であるといえそうであると確認した．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．また，今回検証を行った手法が実際にソフトウェア部品の理解にどれだけ役立てられるかを調査する必要がある．

参考文献

[1] 堀貫行，後藤慧：“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.

[2] 間瀬尚哉，各務秀人，澤井政斉：“利用先や利用元の部品の共通性に基づくソフトウェア分類手法の評価”，南山大学理工学部 2017 年度卒業論文，2018.

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

[4] 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.

[5] 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.