ソフトウェア外で定義された部品の利用の共通性に基づくソフトウェア部品分類手法

2015SE005 遠藤智規　2015SE015 平松芳貴　2015SE040 川村駿弥
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，類似度をもとにソフトウェアを分類することで，機能や役割が似ていると思われる部品を抽出する手法 [1] に対し，適合率および再現率の観点から評価を行い，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，含まれるべき部品の多くを含んでいたことを確認した [2]．しかし，どの部品も利用していない部品や，利用関係が他の部品と一致しない部品がある一定数存在し，それらは分類の対象外となるので，分類対象となる部品数を増やす仕組みが必要である．
本研究では，ライブラリなどのソフトウェア内で定義された部品以外の利用状況も分類に利用すれば，今まで対象外だった部品も分析対象とすることができると考えた．部品毎に，どのライブラリ中の部品を利用しているかの情報を抽出する．その利用関係の一致度に基づいて階層的クラスタリングを行い，部品を分類する．評価実験を行い，分類結果がどのような観点でソフトウェア部品を分類していたかを検証する．さらに改良方法を考察し追加実験を行い，改良方法によって精度が向上するかを調査する．

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 ソフトウェアを分析対象とする．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，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．本研究では，ソフトウェアの中でソースコードが定義されている部品をソフトウェア内で定義された部品，ライブラリのようにすでにソフトウェア外で定義されている部品をソフトウェア外で定義された部品とする．

2.2　利用関係に基づく分類手法

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，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 [5] 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．
我々は，ソースコードから，クラス間の利用関係を抽出し，それをソフトウェア理解支援に役立てる方法を提案している．ある部品においてある機能を実現しようとする場合，他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．堀らは [1] で，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（もしくは利用元）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，その類似度をもとにソフトウェアを分類することで，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．間瀬らは，[1] の手法での分類結果が類似した部品をどれだけ含むことができたかを調査することにより，適合率，再現率の観点から評価を行った [2]．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，含まれるべき部品の多くを含むことができた．

3　ライブラリなどのソフトウェア外の部品への利用関係に基づく分類手法

3.1　問題点と提案するアプローチ

[1] では，ソフトウェア内で定義された部品をどのように利用しているかに基づいて評価を行ったが，この手法の場合，どの部品も利用していない部品や，利用関係が他の部品と一致しない部品がある一定数存在し，それらは分類の対象から外れてしまう．それらの部品の数は，表 1 で示すように全体の 5 %〜60 %にわたる．ライブラリなどのソフトウェア外で定義された部品の利用状況も分類に利用すれば，分類対象となる部品の数が増えると考えられる上に，ソフトウェアをまたいだ部品の分類もできると考えた．
本研究では，部品毎に，どのソフトウェア外で定義された部品を利用しているかの情報を抽出する．その利用関係の一致度に基づいて，部品対ごとに距離を求め，階層的クラスタリングを行い，部品を分類する．評価実験を行い，分析対象となる部品の数が増えるかを確認するとともに，分類結果がどのような観点でソフトウェア部品を分類してい

たかを示すことで，[1] の手法で得られる部品群との違いを調査する．これにより，提案手法がどのように分類に利用可能かについて考察する．

表 1　ソフトウェア内で定義された部品への利用関係の一致度で分類したときに分類されない部品の割合

ソフトウェア名	対象外の部品	部品数	割合	ソフトウェア名	対象外の部品	部品数	割合
flazr	1	27	3.7 %	Jaxen-core	6	163	3.7 %
Jackcess	11	259	4.2 %	jid	4	91	4.4 %
Bacnet4j	16	324	4.9 %	gogui	26	451	5.8 %
jasmin	6	88	6.8 %	jgraphx	30	440	6.8 %
classycle	7	92	7.6 %	ant	45	521	8.6 %
Domination-install	52	284	18.0 %	JSAP	13	69	19.0 %
jfmftpfs	1	5	20.0 %	junit	23	87	26.0 %
kalender	14	50	28.0 %	jlgui	21	70	30.0 %
jmimemagic	6	10	60.0 %				

3.2　分析手順

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，1 で入手した利用関係のうち，ソフトウェア外で定義された部品への利用関係のみを入手し，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，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部品間の関連性を調査する．

3.3　類似度と部品間の距離の定義

ある部品 A が利用しているソフトウェア外で定義された部品の集合を O_A，ある部品 B が利用しているソフトウェア外で定義された部品の集合を O_B としたとき，それらの集合間の類似度 $sim(O_A, O_B)$ を Jaccard 係数を用い，以下のように定義する．

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

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

$$dist(O_A, O_B) = 1 - sim(O_A, O_B)$$

4　評価実験

4.1　実験の内容と評価項目

jlgui を対象として，提案手法を適用する．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．実験では群平均法で階層的クラスタリングを適用し，樹形図を得る．得られた樹形図に対して，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. ソフトウェア外で定義された部品の利用の一致度で分類した場合，分析対象となる部品数が増えるか?
   分析結果として得られた樹形図において，根の付近で結合している部品は分析対象とならない部品である．そのような部品の数をかぞえ，ソフトウェア内で定義された利用関係で分類した場合と，分析対象外となる部品数を比較する．これにより提案手法によって分析対象となる部品数が増えるかを調査する．
2. 樹形図において，まとまりになっている部品について，それらの部品間に関連性はあるのか?
   分析結果において得られた部品群それぞれについて，部品群中の部品の関連性，部品群内の部品の類似性を調査する．関連性においては，共通で利用している部品や多くの部品が共通して利用している部品を調べ，それらが意味付けに役立っているかを調査する．類似性においては，適合率で調査し，部品群内の部品の類似性を調査する．これにより提案手法において得られた部品集合がもつ特徴を調査し，また部品群の部品同士に関連性が確認できるかを調査する．

4.2　類似性の判断基準

[2] で用いた類似性を確認するための基準を用いて，適合率の分析を行った．適合率の分析では，部品群内の部品を 1 つ無作為に選び，その部品を基準として部品群内の部品が，どれだけ基準を満たしたかを求める．具体的には，部品群内で A とその基準で関連する部品数を，部品群中の部品数で割った値を適合率とする．

基準 1　部品の派生元や実装インタフェースが同じである．

基準 2　利用先の 50 %以上が同じである．

基準 3　部品群を 1 つのグループとみたときに，部品の役割や扱う対象から機能群を構成している．

基準 4　同一のシグニチャのメソッドが複数存在する．

基準 5　同じパッケージに所属している．

基準 6　ファイル名の一部が一致している．

基準 7　コードクローンを有している．

4.3　実験の結果

提案手法をもとに jlgui 内の部品を階層的クラスタリングで分類した結果を図 1 に示す．以下ではリサーチクエスチョンに沿って分析結果を紹介する．

1. ソフトウェア外で定義された部品の利用の一致度で分類した場合，分析対象となる部品数が増えるか?
   ソフトウェア内で定義された部品の利用関係で分類した場合は 31 %の 22 個の部品が解析の対象とならなかったが，各部品のソフトウェア外で定義された部品

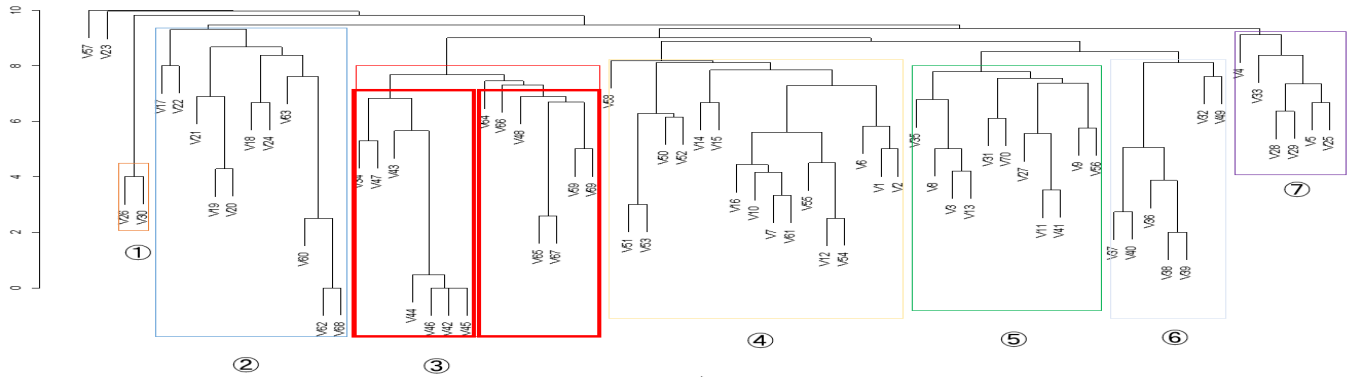

図 1　ソフトウェア外の部品の類似度で作成した樹形図

への利用関係で分類した場合は，1 個の部品のみ対象とならなかった．ライブラリなどの外部の部品はどの部品も利用しており，分析対象を増やすことはできると考えられる．

2. 樹形図において，まとまりになっている部品について，それらの部品間に関連性はあるのか?
   樹形図からは，図 1 で示す 8 個の部品群が抽出できた．表 2 にそれらの詳細を示す．部品群 4 では，共通して利用している部品が，java.lang.Object と java.lang.String であった．この 2 つの部品はどの部品でも使われていると考えるので，分析対象にしてしまうと関連性がない部品でも，共通として捉えてしまうので，部品群 4 は関連性が見られなかった．他の部品群では，使用しているライブラリの部品との関連性はある程度確認できた．部品群 1，3-1，3-2 は多くの基準で高い割合を示したが，部品群 2，4〜7 ではそれほど高い割合を示さなかった．特に部品群 4 ではほとんどの基準で低い割合を示していた．ソフトウェア内で定義された部品への利用関係で分類した場合と比べると，あまり良い結果にはならなかった．ライブラリのどのクラスを利用しているかに基づいた分類となっており，具体的に何を行っているかで分類しているが，その分類は部品の役割より細かい分類である動作に関して分類しており，意味合いが少し異なると考えられる．

5　考察

5.1　手法の改良の方向性について

jlgui に対する適用結果からは，ソフトウェア外で定義された部品への利用関係で分類した場合，分析対象外の部品を少なくすることができるが，その中で関連性は見出すことができるとしても，部品の中で実際に実行している内容に基づいた分類となっており，全体的な役割はあまり考慮できていないと推測した．また，その部品の主となる部品の機能を把握することが求められるが，汎用的な部品の利用はその目的では無価値である．汎用的な部品を共通部品としている部品群では，共通性を導出できなかった．これらを踏まえて，以下の 2 つの改良案を提案する．

改良案 1：汎用的な部品を定義し，それらの部品の利用を分析の対象外として，類似度を計算する．これにより，汎用的な部品を共通して利用している部品群を分解できる．

改良案 2：どのクラスを利用しているかの分類ではより細かい動作に関する分類となった．そこで，解析の粒度をあらくして，どのパッケージの部品を利用しているかに基づいて，類似度を計算し，分類する．得られる結果の粒度があらくなることで，目的に沿った情報が得られることを期待している．

5.2　改良案 1 を適用した結果

汎用的な部品は，ソフトウェア理解の助けにならないと考えて，類似度の計算から除外し，分類を行った．その結果，図 1 で示す部品群 4 の部分が分解され，部品群に散らばっていったが，その他の部品群はあまり影響を受けなかった．部品群ごとに除外前の部品群の適合率と比較しても，結果の向上は見られず，汎用的な部品を分析の対象から外しても効果的でないことを確認した．

5.3　改良案 2 を適用した結果

ソフトウェア外の利用部品のパッケージ名の一致度で分類した場合の樹形図を図 2 で示し，部品群ごとの意味付けと適合率を表 3 に示す．ソフトウェア外の利用部品の一致度で分類した場合の樹形図と比較すると，大きく変化したことがわかる．適合率の観点からは，やや低下する傾向が見られるが，関連性はあっても今まで結合できていなかった部品が，部品群の一部となっている事例を確認した．例えば，図 1 の部品群 4 に存在した ControlCurve と部品群 7 に存在した NaturalSpline が，改良案 2 を適用した結果，図 2 の部品群 10 で同じまとまりになった．その他には，1 つの部品群が複数の関連する部品群から成り立っている事例も見られるようになった．このように，改良案 2 を適用することは，大まかな役割毎に分類することを目的とした場合，有効であると考えられる．

表 2 部品群ごとの意味付けとソフトウェア外の部品との関連と適合率

部品群	部品数	代表的な部品	どのような観点で一致しているか	共通して利用している代表的な部品は何か (部品数)	共通利用部品の役割	基準 1	2	3	4	5	6	7
1	2	DragAdapter,PopupAdapter	中継機能をもつソフトウェア	java.awt.event.MouseAdapter,java.awt.event.MouseEvent(3)	Mouse	2/2	2/2	2/2	2/2	2/2	2/2	×
2	11	ActiveJIcon,PreferenceItem	GUI で使われる	javax.swing.JPanel,javax.swing.Icon,javax.swing.border.Border (9)	GUI	×	6/11	7/11	6/11	7/11	7/11	×
3-1	7	UrlDialog	システム側からのメッセージとして扱う	javax.swing.JButton,javax.swing.BoxLayout,java.awt.GridBagLayout (25)	GUI	5/7	5/7	7/7	5/7	7/7	7/7	×
3-2	7	Preferences,SystemPreference	設定画面などを扱う	java.awt.GridBagLayout,java.awt.GridBagConstraints(13)	GUI	5/7	7/7	7/7	4/7	7/7	6/7	×
4	17	なし	なし	java.lang.Object,java.lang.String (6)	関連はない	×	×	6/17	×	2/17	2/17	×
5	11	PlayerUI,EqualizerUI	UI を扱う	javax.swing.SwingUtilities,org.apache.commons.logging.LogFactory (22)	UI	4/11	5/11	5/11	×	×	×	×
6	7	TagInfo,APEInfo,SkinLoader	読み込む際に扱う	java.lang.Object,java.lang.String,java.net.URL (5)	URL を読み込む処理	4/7	4/7	5/7	4/7	5/7	5/7	×
7	6	Taftb,ImageBorder	画像など取り扱う	java.awt.Image,java.awt.Graphics (3)	画像などの記憶	×	2/6	4/6	2/6	2/6	2/6	×

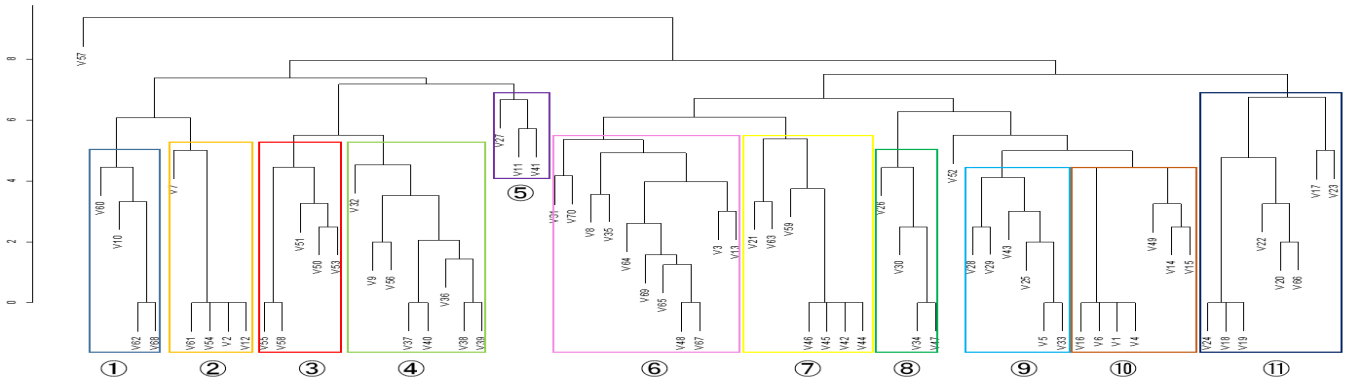

図 2 ソフトウェア外の部品への利用パッケージの類似度で作成した樹形図

表 3 部品群ごとの意味付けと適合率（ソフトウェア外の部品への利用パッケージの一致度に基づく分類手法の場合）

部品群	部品数	代表的な部品	どのような観点で一致しているか	共通して利用している代表的なパッケージは何か	基準 1	2	3	4	5	6	7
1	4	EmptyPreference,OutputPreference	データの保存で使われる	java.lang	3/4	3/4	3/4	3/4	3/4	3/4	×
2	5	PlaylistItem,NodeItem	GUI で使われる	java.lang	×	×	2/5	×	×	2/5	×
3	5	FileNameFilter,FileUtil	ファイルで使われる	java.lang	×	×	4/5	×	3/5	2/5	×
4	8	FlacInfo,TagInfo	読み込む際に扱う	java.lang	5/8	4/8	5/8	5/8	5/8	5/8	×
5	3	PlaylistFactory,TagInfoFactory	GUI で使われる	java.lang	×	2/3	2/3	×	×	2/3	2/3
6	11	Preferences,SkinPreference	設定画面などを扱う	java.lang,java.awt	7/11	6/11	7/11	8/11	4/11	4/11	×
7	7	APEDialog,FlacDialog	システム側からのメッセージを扱う	java.lang,javax.swing	4/7	4/7	4/7	4/7	4/7	4/7	×
8	4	DragAdapter,PopupAdapter	中継機能をもつソフトウェア	java.awt	×	×	2/4	2/4	3/4	2/4	×
9	6	Taftb,ImageBorder	画像など取り扱う	java.lang,java.awt	×	×	2/6	×	4/6	×	×
10	7	ControlCurve,NaturalSpline	波長で使われる	java.lang	2/7	×	2/7	×	3/7	2/7	×
11	8	ActiveJIcon,ActiveJButton	GUI で使われる	javax.swing	2/8	5/8	7/8	7/8	7/8	7/8	×

6 まとめ

本研究では，ライブラリなどのソフトウェア外で定義された部品への利用関係で抽出し，その利用関係の類似度に基づいて分類する手法を提案した．結果として，分類対象となる部品数を増やせること，手法の利用目的も考慮した場合，パッケージ単位の利用関係で分類する手法が有効であることを確認した．今後の課題として，提案手法の精度向上と共に，提案手法 [1] の手法と組み合わせる方法の実現や，利用関係の局所性を考慮した類似度計算方法の実現などにより，手法全体としての精度向上があげられる．

参考文献

[1] 堀貫行, 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.

[2] 間瀬尚哉, 澤井政斉, 各務秀人, ” 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評価” , 南山大学理工学部 2017 年度卒業論文, 2018.

[3] C.Kruegger, ”Software Reuse” , ACM Computing-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.