利用関係の一致度に基づくソフトウェア部品分類手法
ソフトウェア外で定義された部品の利用関係を加味した手法の実現

2017SE005 青山 季暉　2017SE056 名和 大騎
指導教員：横森 励士

1　はじめに

ソフトウェア部品がどの部品を利用しているかに基づいて，部品対ごとに類似性を計算し，階層的クラスタリングを行ってソフトウェア部品を分類する手法が提案されている．類似度を計算する方法として，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をもとに類似度を計算する方法，ライブラリなどのあらかじめソフトウェア外で定義された部品への利用関係の一致度をもとに類似度を計算する方法が今までに提案されている．それぞれの手法で得られる類似部品群の集合は異なっており，それらの結果を組み合わせる方法も考えられるが，まだ検証されていない．本研究では，ソフトウェア内で定義された部品への利用関係とソフトウェア外で定義された部品への利用関係それぞれについて一致度を計算し，それを類似度として分類に用いる方法を提案する場合，組み合わせた結果において既存手法における部品群がどのように変化するかを追跡し，それぞれの既存手法の結果を補完し，より精練された手法になっているか検証し，組み合わせ手法の有効性を確認する．

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．本研究では 1 つの Java ソフトウェアシステム中のソフトウェア部品を対象に，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た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し，それらの部品への利用関係は区別して扱う．

2.2　利用関係の類似度に基づく分類手法

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，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 [1]．また，Li らは，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 [2].

澤井らは，ソフトウェア部品の利用状況の一致度から，ソフトウェアを階層的クラスター分析で分類することで，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 [3]．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を行うために，適合率，再現率の観点から評価を行った．その結果，適合率，再現率ともに高い割合を示すことができ，実際に類似性を持つ部品として含まれるべき部品の多くを部品群に含んでいたことを確認したが，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一定数存在することも確認した．

さらに，遠藤らは，ライブラリ部品などのソフトウェア外で定義された部品への利用関係を抽出し，その利用関係の一致度に基づいて分類する手法を提案した [4]．結果として，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の類似度で分類すると，分類対象となる部品数が増加し，類似性を持つ部品群として扱うことができる部品の数も増加した．一方で，java.lang.object など汎用的な部品の利用でまとまった部品群は類似性を示さなかった．

3　ソフトウェア内外で定義された部品への利用関係に基づく分類手法

3.1　本研究の動機

ソフトウェアを構成している部品を分類し，その分類を提示することで，ひとまとめに理解すべき部品などを一度に提示することができるなど，利用関係の一致度に基づいた分類手法は，ソフトウェアの保守におけるソフトウェアの理解を支援できる手法だと考えている．

[4] では，ソフトウェア外で定義された部品の利用関係の一致度を用いてソフトウェア部品を階層的クラスター分析を行う方法が提案されている．結果として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と比べて，分類対象となる部品数は増え，より多くの部品に対して，類似性を確認できたが，一部の部品については適切な結果が得られなかった．得られた結果を精練するための方法としてソフトウェア内で定義された部品の利用状況の一致度と組み合わせて分類を行う方法も考えられるが，その有効性についてまだ検証は行われていない．

3.2　研究のアプローチ

これらの手法の結果を組み合わせることで，ソフトウェア内の利用関係で分類した結果を，ソフトウェア外で定義した部品への利用関係で分類した結果によって補完できる

ような手法となることが期待できると考えた．提案手法の効果を確認するために評価実験を行う．

本研究はソフトウェア内で定義された部品の利用状況の一致度，ソフトウェア外で定義された部品の利用状況の一致度からそれぞれ類似度を求め，それらを組み合わせて類似度を計算し，階層的クラスター分析を行う方法を提案する．単独の方法で得た類似部品集合が組み合わせた方法で得た場合にどのように変化するかなどの調査を通じて，提案手法が，組み合わせた方法でどのような位置付けの手法となるかについて考察する．また，提案手法を適用することでそれぞれの既存手法の結果を補完し，既存手法に比べてより精練された手法になっているかを検証する．

3.3 比較の対象

本研究では，比較を行うために，既存の 2 つの方法と提案手法の，計 3 つの方法でそれぞれ距離行列を作成する．それぞれの距離行列を用いて階層的クラスター分析を行って樹形図を入手し，類似した部品の集合を抽出する．

1. (既存手法 1) ソフトウェア内で定義された部品への利用関係のみを考慮する．ある部品 A の利用先部品の集合を L_A，ある部品 B の利用先部品の集合を L_B とする．類似度 $sim(L_A, L_B)$ と距離 $dist(A, B)$ を以下のように定義する．

$$sim(L_A, L_B) = \frac{|L_A \cap L_B|}{|L_A \cup L_B|}$$

$$dist(A, B) = 1 - sim(L_A, L_B)$$

2. (既存手法 2) ソフトウェア外で定義された部品への利用関係のみを考慮する．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 とする．類似度 $sim(O_A, O_B)$ と距離 $dist(A, B)$ を以下のように計算する．

$$sim(O_A, O_B) = \frac{|O_A \cap O_B|}{|O_A \cup O_B|}$$
$$dist(A, B) = 1 - sim(O_A, O_B)$$

3. (提案手法) 1．2．を組み合わせてソフトウェア内外で利用した部品の一致度から類似度を計算する．1．2．の計算式を組み合わせて，類似度 $sim(A, B)$ と距離 $dist(A, B)$ を以下のように定義する．

$$sim(A, B) = \frac{sim(L_A, L_B) + sim(O_A, O_B)}{2}$$

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

4 評価実験

4.1 類似性の判断基準と比較の方法

比較においては，得られた 3 つの樹形図それぞれにおいて，樹形図の葉の部品同士で類似している部品対の集合を求め，それらが樹形図に沿ってどれだけ類似した部品をまとまりとできているかを調査する．類似性を最大限確認できる範囲での部品集合を類似部品群として求める．得られた部品集合における類似点を調査し，比較する．部品間の関連性を確認する際には，3 つの基準を設定し，類似性を判定した．

基準 1 分類された部品の扱う対象が同じである．必要な前処理や後処理が一致しているなど，同種の処理が行われている部品として関連していると判断する．

基準 2 分類された部品の役割が同じである．ある 1 つの機能を実現する部品でまとまり，同じ目的の処理が行われている部品として関連していると判断する．

基準 3 部品の役割や対象，パッケージなどから 1 つの大きな役割に対してその一部を実現する部品群であるとして関連していると判断できる．(共通の利用元を考慮することで，その利用元からの機能群として認識できるものを含む．)

4.2 実験の対象としたソフトウェア

PlotDigitizer と JasperReports Library を実験対象のプロジェクトとして，提案手法を適用した．PlotDigitizer は PDF になっているグラフからデータを数値化するための Java プロジェクトで，80 のソースファイル（部品）で構成されている．JasperReports Library は Java で動作するオープンソースの帳票出力ライブラリで，88 のソースファイル（部品）で構成されている．本研究では，PlotDigitizer の ver.2.6.8 と，JasperReports Library の ver.0.2.1 を対象とし，要旨では既存手法 1 と提案手法の比較を行う．

4.3 各方法で得られた部品群について

4.3.1 PlotDigitizer における部品群

PlotDigitizer を対象とし，提案手法をもとに樹形図から部品群を抽出した結果をそれぞれ図 1，図 2 に示す．さらに図 1 で得られたそれぞれの部品群の部品数，部品例，部品群の共通点を表 1 に示す．

- 既存手法 1 により，7 個の部品群が抽出できた．⑥が基準 1，①②⑦が基準 2，③④⑤が基準 3 を満たしていた．
- 提案手法からは，12 個の部品群が抽出できた．⑪が基準 1，②⑥⑧⑩⑫ が基準 2，①③④⑤⑦⑨が基準 3 を満たしていた．

各手法で得られた部品群の数，類似する部品の総数，1 グループあたりの部品の平均数を表 2 に示す．表 2 より，

類似する部品の総数は提案手法を用いた場合が 1 番多くなった．また，提案手法を用いることで既存手法 2 よりも 1 グループあたりの部品の平均数が増えた．

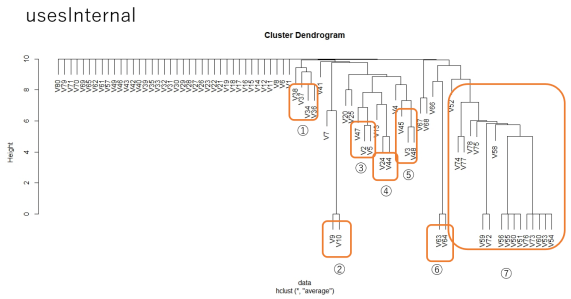

図 1　既存手法 1 で得られた部品群 (PlotDigitizer)

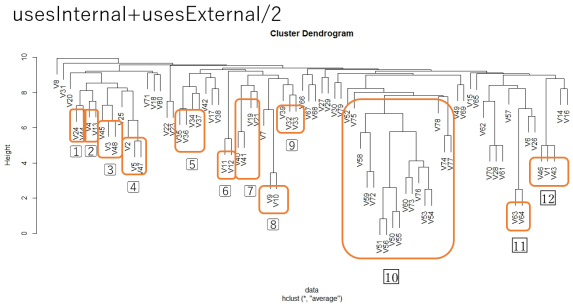

図 2　提案手法で得られた部品群 (PlotDigitizer)

表 1　図 1 で得られた部品群

ID	部品数	部品例	部品群の共通点
1	4	jahuwaldt.tools.tables.FTableReader	表の作成
2	2	com.Data.XMLLoader	XML ドキュメントの読み込み，保存
3	3	jahuwaldt.swing.AutoOptionsDialog	ダイアログの表示
4	2	PDFReader	ファイルの読み込み
5	3	AppWindow	アプリケーションの基盤
6	2	net.roydesign.mac.JD2AppleEventHandlerThunk	ファイルのダウンロードのためのツール
7	17	net.roydesign.ui.JScreenMenu	メニュー画面の表示

表 2　提案手法による分析結果

	得られた部品群の数	類似する部品の総数	1 グループあたりの部品の平均数
既存手法 1	7	33	4.5
既存手法 2	12	41	3.4
提案手法	12	47	3.9

4.3.2　JasperReports Library における部品群

JasperReports Library を対象とし，提案手法をもとに樹形図から部品群を抽出した結果をそれぞれ図 3，図 4 に示す．さらに図 3 で得られたそれぞれの部品群の部品数，部品例，部品群の共通点を表 3 に示す．

- 既存手法 1 により，14 個の部品群が抽出できた．④⑨が基準 1，①⑥⑩⑬が基準 2，②③⑤⑦⑧⑪⑫⑭が基準 3 を満たしていた．
- 提案手法からは，9 個の部品群が抽出できた．⑥が基準 1，①⑤が基準 2，②③④⑦⑧⑨が基準 3 を満たしていた．

各手法で得られた部品群の数，類似する部品の総数，1 グループあたりの部品の平均数を表 4 に示す．表 4 より，類似する部品の総数は提案手法を用いた場合が 1 番多くなった．また，提案手法を用いることで既存手法 1，既存手法 2 よりも得られた部品群の数は少なくなっているが，1 グループあたりの部品の平均数が増えたため，より大きな部品群にまとまった．

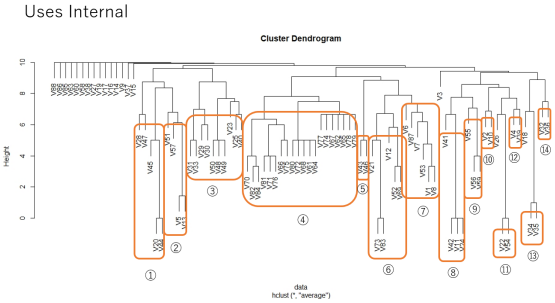

図 3　既存手法 1 で得られた部品群 (JasperReports Library)

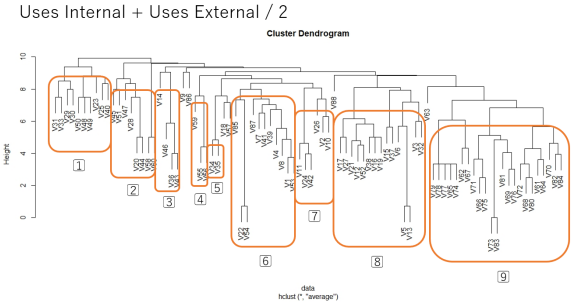

図 4　提案手法で得られた部品群 (JasperReports Library)

表 3　図 3 で得られた部品群

ID	部品数	部品例	部品群の共通点
1	5	dori.jasper.engine.JRPElement.java	エレメントの構成
2	4	dori.jasper.engine.JRBand.java	テキストの処理
3	10	dori.jasper.engine.JRPImage.java	画像処理
4	20	dori.jasper.xml.JRTextFieldFactory.java	データの読み込み，保存
5	2	dori.jasper.engine.JRStaticText.java	テキストの処理
6	6	dori.jasper.xml.JRGroupExpressionFactory.java	データの読み込み，保存
7	6	dori.jasper.engine.JRCalculator.java	計算機能
8	4	dori.jasper.engine.JRDataSource.java	データの処理
9	3	dori.jasper.view.JasperViewer.java	画面表示や出力機能
10	2	dori.jasper.engine.JRCompiler.java	コンパイルを行う
11	2	dori.jasper.engine.JRHorizontalFiller.java	パディングを行う
12	2	dori.jasper.engine.JasperReport.java	データの検索，更新
13	2	dori.jasper.engine.JRPrinter.java	印刷機能
14	2	dori.jasper.engine.JRPText.java	テキストの処理

表 4 提案手法による分析結果

	得られた部品群の数	類似する部品の総数	1 グループあたりの部品の平均数
既存手法 1	14	70	5.0
既存手法 2	12	52	4.3
提案手法	9	82	9.1

5 得られた部品群の比較に関する考察

PlotDegitizer に対して，図 1，図 3 で得られた部品群の変化を表 5 に示す．PlotDegitizer の事例では既存手法 1 の結果を保ちつつ，ソフトウェア外で定義された部品への利用部品を加味することで新たに部品群を得ることができ，有効であると考えられる．次に JasperReports Library に対して，図 3，図 4 で得られた部品群の変化を表 6 に示す．提案手法により，既存手法 1 から部品群が組み換わった事例が多く見られた．結果を見る限りでは，組み換えた後の結果のほうが広い観点から機能的に部品を分類でき，既存手法 1 による分類が不十分であった可能性がある．

この前者と後者の結果の違いは，PlotDigitizer は現行バージョンの 2.6.8 に対して，JasperReports Library はバージョン 0.2.1 とかなり初期のものを分析している．ソフトウェア開発の初期段階では，ソフトウェア内で必要なクラスなどの整備はまだ十分に行われていないことも考えられる．そのため，ソフトウェア内の利用関係だけではまだ分類が不十分である可能性があり，それがこのような結果の違いとして出たと考えられる．このような場合でも，ライブラリ部品の利用などの類似性の観点からより適切な分類結果を導き出せることを示している可能性がある．今後，他のプロジェクトに対して適用し，同様の結果が出るかを調査する必要がある．

表 5 図 1，図 2 で得られた部品群の遷移

図 1 から図 2 への変化	変更点
①→[5]	なし
②→[8]	なし
③→[4]	なし
④→[1]	なし
⑤→[3]	なし
⑥→[12]	なし
⑦→[10]	汎用部品が追加
×→[2]	画像処理を行う部品がまとまった
×→[6]	ログに関する機能を持つ部品がまとまった
×→[7]	フィールドや形式に関する機能を持つ部品がまとまった
×→[9]	表計算に関する機能を持つ部品がまとまった
×→[12]	エディタに関する部品がまとまった

6 まとめ

本研究では，ソフトウェア内で定義された部品の利用状況の一致度，ソフトウェア部品外で定義された部品の利用状況の一致度から類似度を求め，それらを組み合わせて階層的クラスター分析を行う方法を提案した．結果として，それぞれ単独で分類した結果と比べて，分類対象となる部品数が増えた．単独で行った結果を補完する内容となる場合もあるが組み換えが起きた場合もあった．今後の課題として，提案手法の精度の向上と共に，他のプロジェクトにおいてこの提案手法を用いて同じような結果が得られるのか調査する必要がある．

表 6 図 3，図 4 で得られた部品群の遷移

図 3 から図 4 への変化	変更点
①→[2]	[2]の View に関する部品群に View を制御する部品が追加
②→[2][5][8]	[2]にテキストに関する部品が追加 [5]で View のデザインに関する機能を持つ部品群がまとまる [8]の文字の表記に関する部品群に部品が追加
③→[1]	なし
④→[9]	[9]の処理を自動化する機能を持つ部品群に部品が追加
⑤→[3]	[3]のテキストに関する部品群に部品が追加
⑥→[8][9]	[8]の文字の表記に関する部品群と， [9]の処理を自動化する機能を持つ部品群に分かれる
⑦→[6][8]	[6]のデータを保存，計算する部品群と， [8]の文字の表記に関する部品群に分かれた
⑧→[6][7]	[6]のデータを保存，計算する部品群と， [7]のデータを読み込みに関する部品群に分かれる
⑨→[4]	なし
⑩→[7]	データの読み込みに関する部品群に部品が追加
⑪→[6]	データの読み込みや保存，計算などの機能を持つ部品群に追加
⑫→[6]	データの読み込みや保存，計算などの機能を持つ 部品群に命令を実行させる部品が追加
⑬→[5]	なし
⑭→[3][8]	[3]のテキストに関する部品群と， [8]の文字の表記に関する部品群に分かれる

参考文献

[1] 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 (ECOOP2009)，2009，pp.318-343.

[2] 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.

[3] 各務秀人，間瀬尚哉，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

[4] 遠藤智規，平松芳貴，川村駿弥：”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”，南山大学 理工学部 2018 年度卒業論文，2019.