利用関係の一致度に基づくソフトウエア部品分類手法
ソフトウエア更新によって分類結果がどのように変化するかについての考察

2017SE002 安藤勇人　2017SE072 白山遼祐　2017SE087 竹市豊
指導教員：横森励士

1　はじめに

我々の研究グループでは，利用関係の一致度に基づいてソフトウエアを分類する手法を提案しており，保守におけるソフトウエア部品の理解の効率化を支援できると考えられる．過去に行われた研究では，利用関係の類似度に基づき，得られた部品群内に関連があるかどうかについての評価のみを行っており，ソフトウエアの進化を分析するための評価は行われていなかった．本研究では，ソフトウエア部品を分類する手法が保守工程におけるソフトウエアの成長の仕方の分析にも用いることができるかを目的とし検証する．リリースごとのソフトウエアの成長と利用関係の変化の関係を調査し，類似部品群の関係の変化からソフトウェアの進化を分析する手法の確立を目的とする．

2　背景技術

2.1　ソフトウエア部品と利用関係

一般にソフトウエア部品とは，その内容をカプセル化したうえで，環境においてそれらを交換可能な形で実現したシステムモジュールの一部である．本研究では，開発者が再利用を行う単位として.java のファイルをソフトウエア部品とみなす．ソフトウエアは複数のソフトウエア部品で構成されると考えることができ，継承，変数の宣言，インスタンスの作成，メソッドの呼び出し，フィールド参照など「ある部品が他の部品を利用する」,「他の部品からその部品が呼び出される」などといった関係を定義して部品グラフ上で表現することができる．

2.2　利用関係の類似度に基づくソフトウエア部品分類手法と分類手法の特性について

ある部品においてある機能を実現する場合，他の部品で提供されている機能やライブラリを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，ソフトウエアの利用先や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，その類似度を元にソフトウエ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．実験では，利用先部品が多く一致している部品同士は関連性を強く認識できる部品の集合であることが確認できた．橋本らは [2] で，部品群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を調査した．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エアの分類結果が適切であることを確認した．遠藤らは [3] で，ライブラリなどのソフトウエ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，[1] の手法より分析対象となる部品数が増えるかを確認した．結果として，[1] で関連性はあっても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．

[1] の手法は，その部品が利用している「ソフトウエア内で定義された部品の集合」の一致度に基づく類似度を利用する．ソフトウエア内で共通の部品を多く利用している部品同士の場合，類似度が高くなる．この類似度に基づいて距離行列を作成し，クラスタリングを行った場合，樹形図上の葉に位置する部品同士は高い類似性を示した．一方で利用部品が無い，一致しないなどの理由で，分類対象とならない部品が多くみられることが分かった．以降，uses internal による分類と呼ぶ．

[3] の手法は，その部品が利用している「ソフトウエア外で定義された部品の集合」の一致度に基づく類似度を利用する．ソフトウエア外で共通のライブラリ部品を多く利用している部品同士の場合，類似度が高くなる．この類似度に基づいて距離行列を作成し，クラスタリングを行った場合，分類対象となる部品の数は増加した．樹形図上の葉の部分に位置する部品同士の多くは高い類似性を示していたが，汎用的な部品の利用によって結びついた部分が一定数存在し，その部分は共通性を示さなかった．以降，uses external による分類と呼ぶ．

2.3　バージョン間の利用関係の変化

亀井らは [4] で，構造が複雑化したソフトウエアを理解することは容易ではなく，どの部分の利用関係が複雑で，またどの部分が変更にコストがかかるか，という情報を提供することは重要であると考えた．バージョン間のソフトウエア部品間の利用関係がどのように変化したかという情報を視覚的に提供することで，開発者がそれらの情報を共有し，多くの参加者がソフトウエア全体像の変化を把握しやすくなると考えた．[4] では，バージョン間のソフトウエア部品間の利用関係がどのように変化したかを集計して表示するシステムを試作し，実際のオープンソースプロジェクトに対して適用した．それらの情報を共有することを通じて，多くの参加者がソフトウエア全体像の変化を把握しやすくなることを示した．

3 ソフトウエア更新によるソフトウエア部品の関係変化の調査

3.1 研究の目的・動機

我々の研究グループでは，利用部品の一致度に基づいてソフトウエアを分類する手法を提案しており，得られた分類結果が対象となる一つのソフトウエア部品を機能面から分類できていたことを示しており，保守におけるソフトウエア部品の理解の効率化を支援できると考えられる．今までに行われた実験は，分類した結果の精度を評価するものでソフトウエアの進化を分析するための評価は行われていなかった．横森らは [5] で，ソフトウエアの機能追加のモデルは，開発を通じてソフトウエアの構造が複雑になると構造が整理されるようなソフトウエア更新が行われることを示した．本研究では，ソフトウエア部品を分類する手法が保守工程におけるソフトウエアの成長の仕方の分析にも用いることができるかを目的とし検証する．

バージョンごとに利用先などの情報の一致度を用いて階層的クラスター分析を行い樹形図を得ることで，得られた部品群がどのように変化したかや，新しく追加された部品が樹形図上でどのように追加されたかを検証することで，分類手法の特性を調査する．リリースごとのソフトウエアの成長との関係を調査し，ソフトウエア理解のアプローチとしての活用方法を考察する．このような実験を通じて知見を得ることで，類似部品群の関係の変化からソフトウェアの進化を分析する手法の確立を本研究の目的とする．

3.2 研究のアプローチ

複数のバージョンを有する Java で開発されたソフトウエアを対象として，得られた各バージョンごと，類似度計算手法ごとに階層的クラスター分析を行う．樹形図からそれぞれ類似部品群を得る．リサーチクエスチョンを以下のように設定し，ソフトウエアの成長の過程において，分類するクラスタリングの手法が利用できるかを考察する．

1. 各バージョンにおいて追加された部品と関連を持つ部品を調査する際にはどのような手法が適切か？
2. 最初のバージョンの部品群はバージョンを経るごとにどのように樹形図上に配置されるか？
   最新のバージョンの部品群はバージョンを遡るごとにどのように樹形図上で配置されるか？

3.3 類似性の判断基準について

本研究ではバージョンごとに [1],[3] の手法を用いて，それぞれ樹形図を得る．得られた樹形図において，葉となる部分から類似性を判定して，類似性を持つ部品が最大限含まれるように類似部品群を求める．類似性の判断基準については，[1] で議論に用いた類似しているかどうかの基準を用いる．クラスター分析を用いて得られた部品群の部品がどのような関連性を持っているかを確認するために 3 つの基準を設定し判定する．

- 基準 1:分類された部品の扱う対象が同じである．
- 基準 2:分類された部品の役割が同じである．
- 基準 3:部品の役割や対象やパッケージなどから 1 つの機能群としてまとめられると考えられる部品群である．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場合も含む．

4 評価実験

SourceForge で公開されている Java を開発言語とした，5 バージョン以上のソースコードが入手できるという条件のもとで，SweetHome3D と LaTeXDraw を分析対象とした．SweetHome3D では住宅を設計し，家具等を配置しながら，3D ビューで間取りを確認できるソフトウエアである．LaTeXDraw は LaTeX 用のグラフィック描写エディタである．

リサーチクエスチョン 1〜2 に対応した実験を評価実験 1〜2 で行う．評価実験 1 では，ソフトウエア更新によって新しく追加された部品が樹形図上でどこに位置するか，得られた部品がほかのどの部品と類似性を持つかなどを調査する．表 1 は SweetHome3D における Ver1.0 から Ver1.5.1 の各バージョンで部品数（.java ファイルの数）がどのように増減したかを示した表である．評価実験 1 では以下の 2 つのバージョンについて調査を行った．ver1.1 では，寸法を編集する機能が追加された．ver1.2.1 では，新しい家具カテゴリの作成によるバグや一部の家具の色や視認性を変更できなかったバグの修正が行われた．評価実験 2 では，最初や最新のバージョンの樹形図から類似部品群を得る．得られた部品群がバージョン更新によって次のバージョンではどのような配置をしていたかを調査する．

表 1 SweetHome3D での各更新における部品数の増減

	総部品数	増えた部品数	減った部品数
ver1.0	82	0	0
ver1.1	85	3	0
ver1.2	79	0	6
ver1.2.1	96	17	0
ver1.3	97	1	0
ver1.3.1	97	0	0
ver1.4	110	15	2

4.1 評価実験 1

4.1.1 SweetHome3D

部品数が大きく増えている ver1.1，ver1.2.1 について調査し，手法を適用したときの樹形図を図 1〜図 4 に示す．図上の赤枠は追加された部品を表し，緑枠は追加された部品と関連をもつ部品の集合として樹形図で認識出来た部品の集合を表す．

- ver1.1（図 1〜図 2）
  uses internal の樹形図では追加された部品全てにおい

て分類されていなかった．uses external の樹形図では追加された部品同士が部品群になることはなかったが，樹形図上で近い場所に分類され，その部品群に関連があることが分かった．

- ver1.2.1（図 3〜図 4）

  uses internal の樹形図では追加された部品 5 つが同じ部品群に分類された．uses external の樹形図では更新履歴の変更内容にあった家具カテゴリに関する部品群を確認することができ，関連のある部品群が多く見られた．

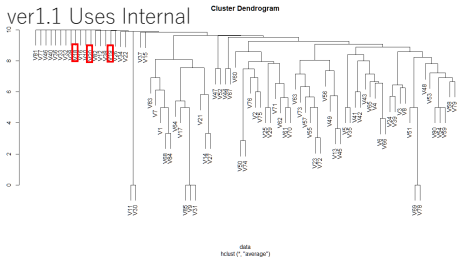

図 1　ver1.1 に対して uses internal で作成した樹形図

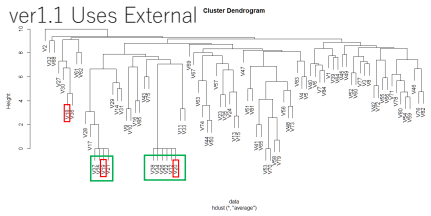

図 2　ver1.1 に対して uses external で作成した樹形図

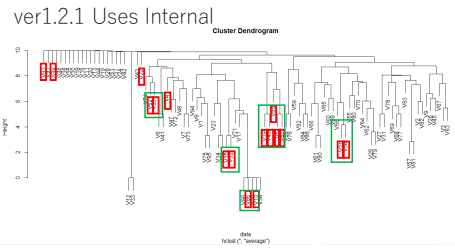

図 3　ver1.2.1 に対して uses internal で作成した樹形図

4.2　評価実験 2

4.2.1　uses internal による分類結果

実験 2 で得られた初期のバージョンの樹形図と，その後の更新で部品群に変化のあった樹形図を図 5 と図 6 に，実験 2 で得られた最新のバージョンの樹形図と，その後の更新で部品群に変化のあった樹形図を図 7 と図 8 に示す．図 5 と図 6 を比較したとき，部品群は保たれているが，一部は分解され，遠い位置に配置されている．図 7 と図 8 を比較したとき，一部の部品群が分解されており，更新の途中で構成された部品群であることが分かる．

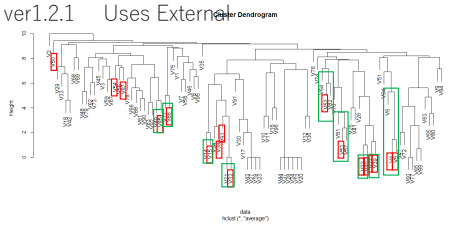

図 4　ver1.2.1 に対して uses external で作成した樹形図

4.2.2　uses external による分類結果

実験 2 で得られた最新のバージョンの樹形図と，初期のバージョンの樹形図をそれぞれ図 9 と図 10 に示す．

得られた樹形図から類似部品群がほとんど分解されずそのまま残っていることが分かる．また最新のバージョンで 5 つの部品で構成されていた類似部品群は，初期のバージョンでは 1 つの部品になっていることを確認した．

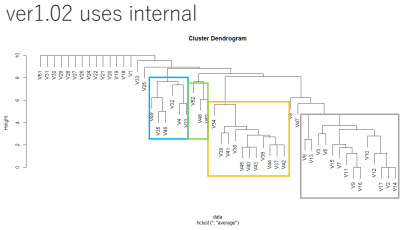

図 5　初期（1.0.2）において得られた部品群（uses internal）

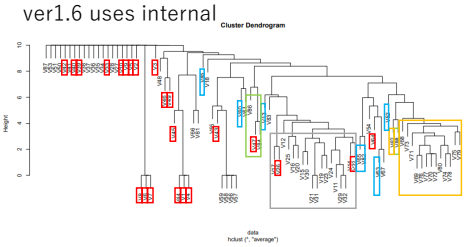

図 6　図 5 の部品群が ver1.6 の樹形図ではどう表されるか

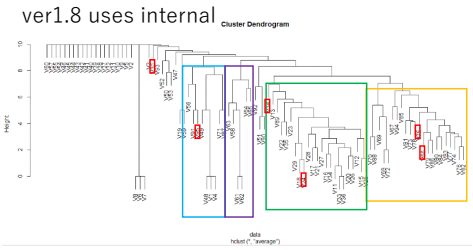

図 7　最新（ver1.8）において得られた部品群（uses internal）

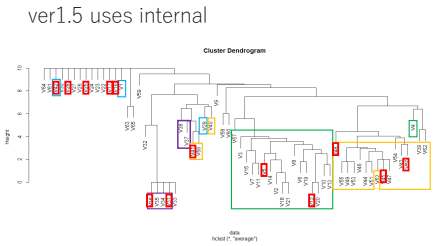

図 8　図 7 の部品群が ver1.5 の樹形図ではどう表されるか

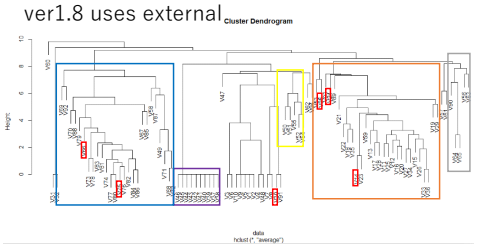

図 9　最新（ver1.8）において得られた部品群（uses external）

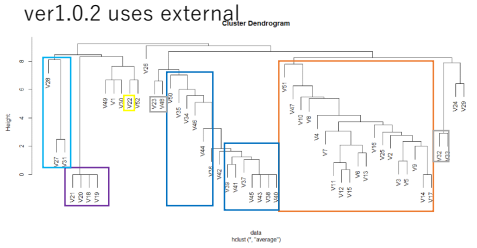

図 10　図 9 の部品群が ver1.0.2 でどう表されるか

5　考察

5.1　リサーチクエスチョン 1 についての考察

実験 1 より，既存のものと同じような機能が追加され，ソフトウエア内に機能を追加する基盤が出来ていて，すでに定義されている部品を多く使用している場合，uses internal で作成された樹形図で見ると関連性が見える可能性が高いと推測できる．機能の拡張や，新しい機能の追加がされたときは，ソフトウエア内の機能を追加する基盤が整っているいないに関わらず，uses external で作成された樹形図は関連性がある程度見えやすいと考えられる．

5.2　リサーチクエスチョン 2 についての考察

uses internal による分類手法では初期のバージョンでは機能を追加する基盤がソフトウエア内でまだ整っていないために，類似部品群の一貫性が低いことが考えられ，バージョンを追うごとに機能を追加する基盤が整い，ソフトウエアが成熟することによって部品間の関連性が高くなっていくと考えられる．またある段階から，部品間の関連性の精度が上がり安定することから，ソフトウエア内の部品の利用関係が整理されソフトウエア内の機能を追加する基盤が整ったのではないかと推測できる．uses external による分類手法では，類似部品群の部品間の関連性がどのバージョン間でも高いことから，類似性を最低限担保するための基盤として利用することでソフトウエアの理解のアプローチとなりそうである．

6　まとめと今後の課題について

本研究ではバージョンごとに利用先などの情報の一致度を用いて階層的クラスター分析を行い樹形図を得ることで，新しく追加された部品が樹形図上でどのように追加されたかや，得られた部品群がどのように変化したかを研究した．結果として，uses internal で作成された樹形図は，ある段階から部品間の関連性の精度が安定し，ソフトウエア内の部品の利用関係が整理されソフトウエア内の機能を追加する基盤が整うと考えられる．ソフトウエア内に機能を追加する基盤が整っており，ソフトウエアがある程度成熟している場合，部品間の関連性が高いという特性や，uses external で作成された樹形図では，ソフトウエアが成長する過程においても，部品間の関連性がある程度高いという特性を確認した．ソフトウエアの進化の過程において実装優先で開発する場合には uses external による手法で分類し，リファクタリングによって再設計される場合には uses internal による手法で分類することが望ましいと考えられる．今後の課題として，ソフトウエア内のどの部品の利用関係が整理されたことによってソフトウエア内の機能を追加する基盤が整っていくかの調査が必要である．

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エ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] 橋本敬太，川瀬史也：“利用部品の共通性に基づくソフトウエア部品分類手法の評価”，南山大学 理工学部 2018 年度卒業論文，2019.

[3] 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 情報理工学部 2018 年度卒業論文，2019.

[4] 亀井雄佑，木下祐太郎，前原一幾：“バージョン間の利用関係の変化を提示するシステムの試作”南山大学 理工学部 2012 年度卒業論文，2013.

[5] Reishi Yokomori,Harvey Siy, Norihiro Yoshida, Masami Noro, Katsuro Inoue : ”Further Considerations about Relationship between Framework and Application Components ”, International Journal of Computer Science and Application ,Vol. 4, Issue. 1 , pp.18-31 , 2015.