

利用関係の一致度に基づくソフトウェア部品分類手法 -パッケージ分類に基づく部品管理における利用方法の考察-

2017SE084 高田晃弘 2017SE075 杉野圭

指導教員：横森励士

1 はじめに

我々の研究グループでは、機能面から類似性を持つ部品を分類する手法として、利用部品の一致度に基づいてソフトウェアを分類する手法を提案している [1]。この手法は、保守においてソフトウェア部品の理解が必要な場面で、類似性を持つ部品をまとめて提示することができるなど、保守におけるソフトウェア理解を支援する方法として考えられる。過去の研究では、既存の様々な分類手法と比較を行い、手法間の特性を調査したが、一番特性が似ていたのはパッケージによる分類であった。[1]の手法によって得られた部品群も所属パッケージに従っていたが、細かい分類結果の違いがソフトウェア部品の管理にどのように役立つかはまだ詳しく調べていない。問題の修正だけでなく、様々な要素への対応として機能追加が行われる保守工程において、利用関係の一致度により類似部品を求める手法をパッケージ分類にどのように活かすかについての知見を得たいと考える。本研究では、このような知見を目的として、ソフトウェア部品を提案手法で分類した結果が、パッケージによる分類結果上でどのように表されるかについて調査を行う。具体的には、パッケージをまたがって得られる部品群について、その部品群がどのような理由で得られているかを調査することで、パッケージを横断している要素としてどのようなものが考えられ、それらが一般的に知られているものについてなのかを調査する。パッケージ内を分類する要素については、分類結果がパッケージ内を更にもどのように分類しているかや、分類結果がそのあとのバージョンのパッケージ構造においてどのように配置されているかなどを調査することで、ソフトウェアが成長していく中で、その後の部品の管理でどのように利用できるかについて調査する。

2 背景技術

2.1 java におけるパッケージ

Java におけるパッケージとは、名前空間を用いて大量のクラスを管理する機構である。それぞれのクラスは「パッケージ名.クラス名」という形で完全限定名によって特定される。パッケージを明示していないクラスファイルはデフォルトパッケージという形で取り扱う。開発者は同じカテゴリーに属しているクラスや類似した機能を提供するクラスを1つにまとめるために、パッケージを用いてクラスを管理する。所属するパッケージの違いによって、アクセスを制限するためにも利用される。import 文を使って他のパッケージに属するクラスを利用できる。

2.2 利用関係に基づくソフトウェア部品分類手法

ある部品においてある機能を実現しようとする場合、開発者は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。掘らは利用先に関して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 [3]。1つのJavaソフトウェアシステムを分析対象として、各クラスのソースコードファイルを部品とみなし、部品間の利用関係を部品グラフで表現し、利用関係の一致度をもとにソフトウェア部品を階層的クラスタ分析で分類する方法を提案した [1]。実験では、利用先部品が多く一致している場合は、部品同士の役割が同じであったり、機能群としてまとまりを確認でき、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。

2.3 他の分類手法との比較について

日比野らは [1]の手法で得られた部品群を、コードクローン分析ツールから得られた類似情報や、実装(継承)元と先の関係から得られた類似部品群や、所属パッケージから得られた類似部品群などの既存の手法による分類結果との比較を行った [2]。結果として、コードクローン分析ツールから得られた類似部品群は、コードクローン検出の精度を上げると正しいグループが得られるが、対象部品の数が少なく、ソフトウェア全体の部品からは情報を入手できなかった。一方で、精度を下げてたくさんの部品を対象にしても、グループ分けの精度が著しく低下し適切な分類にならなかった。実装(継承)元と先の関係から得られた類似部品群を抽出した場合は、部品群内に機能的関連性が低い部品が含まれ、適切な分類結果ではなかった。所属パッケージから部品群を分類する場合は、システム全体の部品をある程度は分類できており、[1]の手法によって得られた部品群も所属パッケージに従っている事例も多くみられた。ただし、パッケージ内の全ての部品が一つに分類されているわけではなく、パッケージ内の一部のみが部品群となっている場合や、パッケージをまたがる部品群が存在している。

3 パッケージによる分類手法との比較

3.1 研究の動機

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ェ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ソフトウェアの中の部品を機能面から分類できていることを示した。この手法は、機能面で類似性を持つ部品を分類することができており、保守にお

いてソフトウェア部品の理解が必要な場面で、類似性を持つ部品をまとめて提示することができるなど、保守におけるソフトウェア理解を支援する方法として考えられる。過去の研究では、既存の様々な分類手法と比較を行い、手法間の特性を調査したが、一番特性が似ていたのはパッケージによる分類であった。パッケージによる分類というのは開発者がソフトウェア全体を機能面や役割、クラスの目的で分類したものである。開発者は、システム全体の部品を所属パッケージの観点から分類しており、[1]の手法によって得られた部品群も所属パッケージに従っていた。だが、パッケージ内のすべての部品が1つに分類されているわけではなく、パッケージ内の部品の一部のみを部品群化していたり、パッケージをまたがる部品群も存在していた。ソフトウェア全体を広く分類する方法として、これらの違いがソフトウェア部品の管理にどのように役立つかはまだ詳しく調べていない。

3.2 比較のアプローチ

問題の修正だけでなく、様々な要素への対応として機能追加が行われる保守工程において、利用関係の一致度により類似部品を求める手法をパッケージ分類にどのように活かすかについての知見を得たいと考えている。本研究では、このような知見を目的として、ソフトウェア部品を提案手法で分類した結果が、パッケージによる分類結果上でどのように表されるかについて調査を行う。具体的には、パッケージをまたがって得られる部品群について、その部品群がどのような理由で得られているかを調査することで、パッケージを横断している要素としてどのようなものが考えられ、それらが一般的に知られているものについてなのかを調査する。パッケージ内を分類する要素については、分類結果がパッケージ内を更にどのように分類しているかや、分類結果がそのあとのバージョンのパッケージ構造においてどのように配置されているかなどを調査することで、ソフトウェアが成長していく中で、その後の部品の管理でどのように利用できるかについて調査する。

3.3 調査項目

以下のリサーチクエスチョンを設定し調査実験を行う。

3.3.1 パッケージ間をまたがる部品群について

- パッケージ間をまたがって得られる部品群が、どのような理由で得られているのか？パッケージを横断する要素として部品群における共通点は一般的に知られたものか？

3.3.2 パッケージ内を分類する部品群について

- 分類結果がパッケージを更にどのように分類しているのか？その後のバージョンのパッケージ構造はどのように変化し、後のバージョンで作成されるサブパッケージは部品群と関連があるのか？

4 評価実験

実際の Java ソフトウェアを対象として、[1]の手法で得られた部品群と、開発者がパッケージ分けを行った結果を比較し、相違点について調査した。実験は、Sudoku for Java - HoDoKu と OmegaT という Java アプリケーションに対して適用実験を行った。HoDoKu は数独パズルを解析するための機能を提供する Java アプリケーションで、Ver2.13 は 130 の java のソースファイル (部品) で構成されている。OmegaT は翻訳 Java アプリケーションで、複数の翻訳対象ファイルの同時処理や同時使用などの機能があり、Ver1.8.1 は 152, Ver3.6.0 は 522 の Java のソースファイル (部品) で構成されている。実験 1 では、HoDoKu を対象として、ソフトウェア内で定義された部品間の利用関係から階層的クラスター分析を行い、類似部品群を得る。パッケージを表現した図上で類似部品群を表現し、比較を行う。所属パッケージをまたがる部品群について着目し、その相違点について調査を行うことで、相違点が何を表しているのか、またどのような活用方法があるのかについて調査する。実験 2 では、OmegaT を対象とし、実験 1 と同様に樹形図を求め類似部品群を得る。Ver1.8.1 と Ver3.6.0 のパッケージ構造がどのように移り変わったかを調査し、Ver3.6.0 までに追加されたサブパッケージが Ver1.8.1 で得られた部品と関連があるか、サブパッケージの元となる部品が存在するかについて調査を行う。

4.1 類似部品群の入手方法と類似性の定義について

得られた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し、最大限得られた部品の集合を類似部品群とする。類似性の判断基準については、[3]での類似性についての議論を参考にして、以下の基準を設定した。

- 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した場合、その利用元からの機能群として認識できる場合も含む。

4.2 実験 1

実験 1 で、HoDoKu の分析により得られた樹形図と部品群を図 1 に示す。図 2, 図 3 では、図 1 で得た部品群をパッケージ分類結果上に示す。得られた部品群の多くはパッケージ分類に沿ったもので、開発者によるパッケージ分類結果を反映できているが、パッケージ内のすべての部品が 1 つの部品群となっているわけではない。複数のパッケージをまたがる部品群として、4 つの部品群を確認できる。以下ではそれぞれの部品群における共通点を示す。

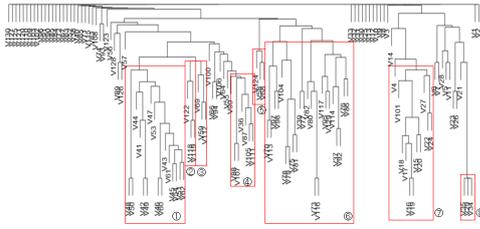


図 1 HoDoKu Ver2.13 から得られた樹形図と部品群

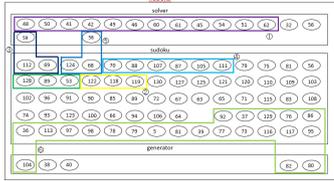


図 2 HodoKu の分析で得られた部品群をパッケージ分類上で表した図 (その 1)

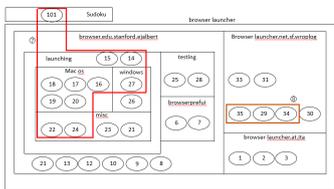


図 3 HodoKu の分析で得られた部品群をパッケージ分類上で表した図 (その 2)

- 部品群 3 は SudokuSet と Chain のクラスの利用で結びついていた . Chain というクラス の概念を用いた解析手法を実現している . 実現している部分が部品群として主パッケージと Solver パッケージにまたがっている .
- 部品群 5 は SudokuSet と TableEntry のクラスの利用で結びついていた . TableEntry というクラス の概念を用いた解析手法を実現している . 実現している部分が部品群として主パッケージと Solver パッケージにまたがっている .
- 部品群 6 は主に options というクラスの利用で結びついていた . イベント処理を行うようなクラスと , Background 処理が行われる generator パッケージが含まれていて , 主パッケージと Generator をまたがる部品群として得られた .
- 部品群 7 は BrowserLaunchingInitializingException というクラスの利用で結びついていた . ブラウザを起動するためにすべき初期化処理での例外の処理の目的で利用しており , このアプリケーション内からブラウザを開くための API を提供する処理の部分で利用されているため , これらの処理を実装するパッケージを

またがっている .

4.3 実験 2

実験 2 で , OmegaT Ver1.8.1 の分析により得られた樹形図と部品群を図 4 に示す . 図 5 , 図 6 では , 図 4 で得た部品群をパッケージ分類結果上に示す . 古いバージョンのパッケージ構造はどのように変化し , 新しいバージョンで作成されるサブパッケージは部品群と関連があるのかを調査するため , Ver3.6.0 までに追加されたパッケージと関連があるかを調査した . glossaries や help , matches のように , 部品群内の 1 つのクラスが関連している事例は見られたが , 共通点がそのままパッケージとして追加されている例は見られなかった . また , 図 7 , 8 で追加されたパッケージから以下の 3 つのパッケージを選び , その機能に関連する Ver1.8.1 におけるクラスを調査した .

- gui.editor が追加された . ver1.8.1 で , Editor に関連した部品は以下で図 9 に青色で示す .
 - org.omegat.util.gui.ExtendedEditorKit
 - org.omegat.gui.filters2.FilterEditor
 - org.omegat.gui.filters2.InstanceEditor
 - org.omegat.gui.main.EditorTextArea
- core.TagValidation が追加された . ver1.8.1 で , Tag に関連した部品は以下で図 9 に緑色で示す .
 - org.omegat.filters3.Tag
 - org.omegat.filters3.xml.XMLInstactTag
 - org.omegat.filters3.mxl.XMLTag
 - org.omegat.gui.TagValidationFrame
- core.dictionaries が追加された . ver1.8.1 で , Dictionaries に関連した部品は以下で図 9 に紫色で示す .
 - org.omegat.core.spellchecker.DictionaryFileNameFilter.java
 - org.omegat.core.spellchecker.DictionaryManager.java
 - org.omegat.gui.dialogs.DictionaryInstallerDialog.java

いずれの場合も , 関連するクラスは樹形図で散らばっており , 類似部品群を構成しなかった .

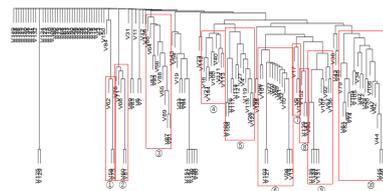


図 4 OmegaT Ver1.8.1 から得られた樹形図と部品群

