メソッド抽出リファクタリングが行われるメソッドの特徴調査

後藤 祥　吉田 則裕　藤原 賢二　崔 恩瀞　井上 克郎

メソッド抽出とは，既存のメソッドの一部を新たにメソッドとして抽出する作業のことである．メソッド抽出は，長すぎるメソッドや，凝集性の低いメソッドを分割するために有効であるとされているが，実際に開発者がどのようなメソッドを対象にしているかは調査されていない．しかし，メソッド抽出作業の支援を行うためには，開発者がどのようなメソッドを抽出の対象としているかを定量的に調査する必要がある．本研究では，オープンソースソフトウェアから，抽出対象となったメソッドのサイズと凝集度が，一般的な値と比較して異なるか調査を行ったところ，多くの場合に有意差が確認できた．

“Extract Method” is a refactoring pattern that extracts a part of an existing method as a new method. Although extract method refactoring is an effective way to decompose long and non-cohesive methods in general, how developers choose methods for “Extract Method” refactoring is still unexamined. For supporting this refactoring, the investigation of it is neccessary. In this study, we investigated the differences of the size and cohesion of methods between methods for this refactoring and general methods in open source software. The result shows significant deliverances in the most cases.

1　はじめに

リファクタリングとは，ソフトウェアの外的な振る舞いを保ったまま内部構造を整理する作業のことであり，プログラムの保守性や可読性の向上を目的として行われる [1]．Fowler は，様々なリファクタリングパターンをまとめており，それぞれのパターンについてその用途や手順を記述している [1]．リファクタリングはソフトウェアの保守性を向上させるが，大規模ソースコード中からリファクタリングの対象を特定する作業は大きな労力を要する．そのため，リファクタリング作業者を支援するための手法やツールが多く提案されている [2] [6]．

近年，開発者が行うリファクタリングの実態を調査し，その結果に基いて支援手法を考案する研究が行われている [3] [7]．既存研究では，各リファクタリングパターンが適用される頻度や Eclipse のリファクタリング機能の利用状況などが調査されているが，リファクタリング対象となったソースコードが持つ特徴の定量的な調査は確認されていない．一般的に，凝集度が低い，もしくはサイズの大きいメソッドがリファクタリングの対象となるとされているが，実際の開発プロジェクトにおけるリファクタリングが，それら尺度に基いて行われているかどうか明らかではない．そのため，それら尺度に基いてリファクタリング対象を推薦すべきであるかどうか判断することが難しい．

本研究では，頻繁に行われるリファクタリングの 1 つであるメソッド抽出 [3] に着目し，抽出対象となったメソッドの特徴を調査した．具体的には，メソッドのサイズや凝集度がリファクタリング対象を発見する基準として有用かどうかを調査するために，抽出対象となったメソッドのサイズおよび凝集度を計測し，

An Investigation into the Characteristics of Methods for Extract Method Refactoring

Akira Goto, Eunjong Choi, Katsuro Inoue, 大阪大学大学院情報科学研究科, Graduate School of Information Science and Technology, Osaka University.

Norihiro Yoshida, Kenji Fujiwara, 奈良先端科学技術大学院大学情報科学研究科, Graduate School of Information Science, Nara Institute of Science and Technology.

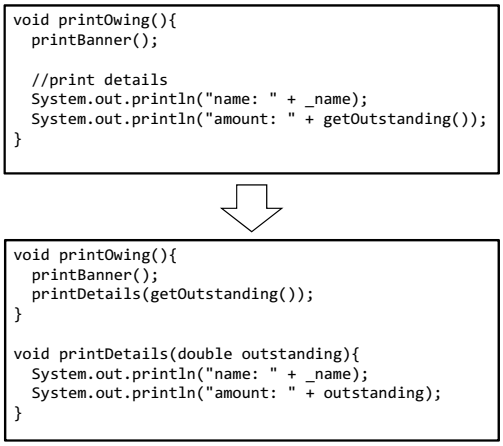


図 1　メソッド抽出リファクタリング [1]

一般的なメソッドのサイズおよび凝集度との比較を行った．比較の結果，抽出対象となったメソッドと，一般的なメソッドのサイズおよび凝集度には有意差が存在することがわかった．

2　背景

2.1　メソッド抽出

メソッド抽出は，既存のメソッドの一部を抽出し，新たにメソッドとして定義するリファクタリングである．図 1 にメソッド抽出の例を示す．図 1 の例において，2 つの文が抽出され，新たに *printDetails* メソッドとして定義されている．抽出元の *printOwing* メソッドでは，抽出部分のソースコードが *printDetails* メソッドの呼び出し文に置換されている．

メソッド抽出は，長すぎるメソッドや，凝集性の低いメソッドの分割を目的として行われるとされている．しかし，実際に長いメソッドが抽出の対象となっているのかどうかや，数値的な尺度などを定量的に調査した研究は著者らの知る限り存在しない．

2.2　藤原らのリファクタリング検出ツール

リファクタリング検出ツールとは，ソフトウェアの開発履歴から，どのリファクタリングパターンが，どの部分に適用されたかを検出するためのツールである．リファクタリング検出ツールを用いて，リファクタリングの事例を収集することで，リファクタリングの頻度やリファクタリング対象とされるソースコードの特徴など様々な調査を行うことが可能となる．

藤原らは，構文情報を付加したリポジトリを利用して，メソッド抽出リファクタリングの検出を行うツールを提案している [9]．藤原らのツールでは，コミット間での変更情報を元にメソッド抽出が行われたと推定される箇所を特定し，抽出元のメソッド，メソッド抽出によって新たに作成されたメソッド，抽出元のコード片と新たに作成されたメソッドの類似度を出力する．この類似度は 0 から 1 の値であり，類似度が高いほどメソッド抽出が行われた可能性が高い．この手法では，メソッド抽出リファクタリングのみを対象としているが，検出精度が優れており，大規模な履歴に容易に適用可能であるという利点がある．本研究では，比較調査のために多くのリファクタリング事例が必要であるため，藤原らのツールを用いてメソッド抽出事例の収集を行った．

3　調査方法

3.1　調査する特徴

メソッド抽出は長すぎるメソッドや，凝集性の低いメソッドを分割するために有効であるとされている．本研究では，メソッドのサイズと凝集性を，それぞれメソッドの文数と凝集度という尺度で計測を行い，抽出対象となったメソッドが持つ特徴を調査する．文数と凝集度について，具体的にどのようにして計測するか説明する．

1. NOS(Number Of Statements)
   メソッドのサイズの計測方法については，単純に行数を測るものや，空白行やコメントを除いて測る方法などが存在する．メソッド抽出はメソッド中の文に対して行われるリファクタリングであり，メソッド中のコメントや空白行の数とは大きな関連がないと思われる．そのため，本研究では，メソッドのサイズをコメントと空白行を除去したメソッドの本体中の文数とした．
2. 凝集度
   凝集度とは，モジュールが機能的にまとまったものかどうかを表す度合である [5]．本研究では，メソッドの凝集度を測るメトリクスとして，Weiser

表 1　調査対象ソフトウェア (括弧内は凝集度が計測できたメソッドの内数)

ソフトウェア	調査対象期間	メソッド抽出事例数	比較用バージョン	メソッド数
jEdit	1998/9/27 - 2012/1/30	490(286)	4.5.0	6275(2114)
Apache Ant	2000/1/13 - 2012/5/23	659(302)	1.8.4	9123(2167)
ArgoUML	1998/1/27 - 2011/12/15	704(322)	0.34	13840(4457)

が提案したスライスベースのメトリクス[8] のうち，Ott らの実験[4] によって有用性が確認されている Tightness，Coverage，Overlap の 3 つを使用する．これら 3 つのメトリクスの定義を以下に示す．式において，M をメソッド，$length(M)$ を M の文の数，V_o を M における出力変数の集合，SL_x を変数 x を起点にした後ろ向きスライス，SL_{int} を V_o 中の全変数に対する後ろ向きスライスの積集合とする．

$$Tightness(M) = \frac{|SL_{int}|}{length(M)}$$

$$Coverage(M) = \frac{1}{|V_o|}\sum_{x\in V_o}\frac{|SL_x|}{length(M)}$$

$$Overlap(M) = \frac{1}{|V_o|}\sum_{x\in V_o}\frac{|SL_{int}|}{|SL_x|}$$

メトリクスの計算に必要となるメソッドの出力変数 V_o は，Tsantalis らの定義に従い，メソッドの返り値と，メソッドの本体とスコープが一致する変数とした[6].

3.2　調査対象

本研究の調査対象として jEdit，Apache Ant，ArgoUML の 3 つのオープンソースソフトウェアを選択した．まず，調査対象のソフトウェアに藤原らのツールを適用し，メソッド抽出事例の検出を行った†1．調査対象の詳細を表 1 に示す．表 1 において，メソッド抽出事例数は，藤原らのツールを使用して検出された調査対象期間内で行われたメソッド抽出のうち，テストメソッドを除いたメソッドの数である．また，比較用バージョンは，抽出対象となったメソッドとの特徴の比較を行うための，調査期間における最新のリリースバージョンを意味しており，メソッド数は，比較用バージョンに存在する全メソッドから，テストメソッドとメソッドの本体が空のメソッドを除いた数である．表 1 中のメソッド抽出事例数とメソッドの数の括弧内数は，これらのメソッドのうち凝集度の計測ができたメソッド数を示している．本研究で用いるスライスベースの凝集度メトリクスは，メソッド内にスライスの起点となる変数が存在しない場合，凝集度の計算ができない．そのため，凝集度の比較においては，凝集度が計算できなかったメソッドは除外した．

†1　藤原らの研究[9] において，類似度の閾値が 0.3 の時，検出精度が最も高いと述べられているため，本研究でも同一の閾値を利用し，出力結果から閾値以上の類似度をもつものを調査対象として利用した．

4　調査結果と考察

各対象ソフトウェアについて，計測した NOS と凝集度の分布を表した箱ひげ図を図 2,3 に示す．以降の結果において,「リリース版」は比較用のリリースバージョンに対する結果を意味しており,「抽出」はメソッド抽出の対象になったメソッドに対する結果を意味している．また図 3 において，T, C, O はそれぞれ Tightness, Coverage, Overlap の 3 つのメトリクスを意味している．

さらに，抽出対象となったメソッドと比較用バージョンに存在するメソッドについて，計測した特徴に有意差が存在するかどうか，マンホイットニーの U 検定を用いて検定を行った．表 2，3，4 に，NOS と凝集度の中央値と検定の結果得られた p 値を示す．

表 2，3，4 より，jEdit における Coverage 以外の全てにおいて，有意水準 0.05 以下で有意差があるという結果になった．これらの結果から，今回対象としたソフトウェアにおいては，メソッドの NOS が多く，凝集度が低いメソッドが抽出の対象となっていると言える．図 2 を見ると，NOS については，抽出対象と

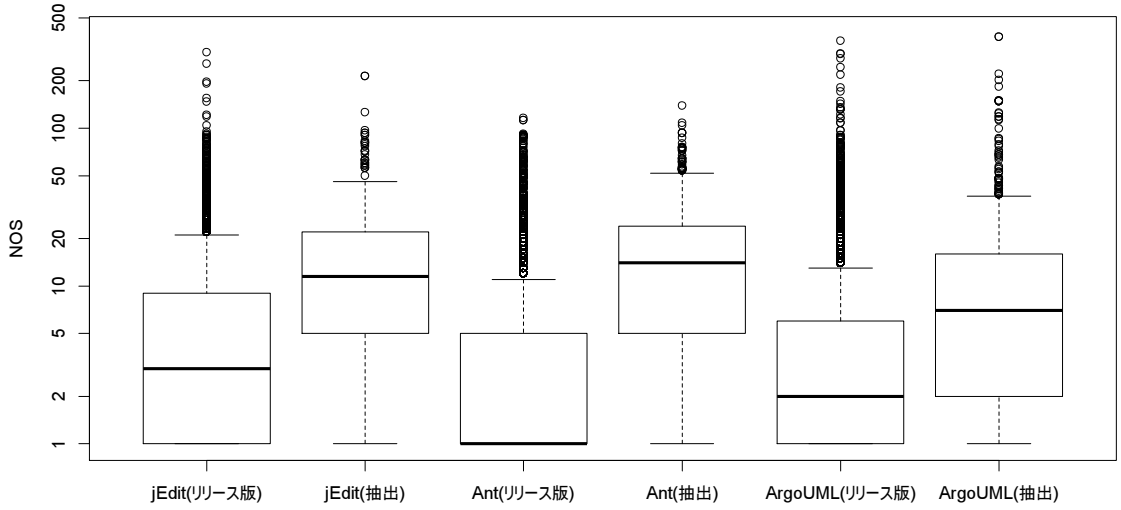


図 2　NOS 分布

なっているメソッドの多くは，約 5 文から 20 文程度であることがわかった．分布に有意差があり，抽出対象となる数値的な基準がわかったことから，NOS はメソッド抽出候補の推薦において有用な尺度であると考えられる．

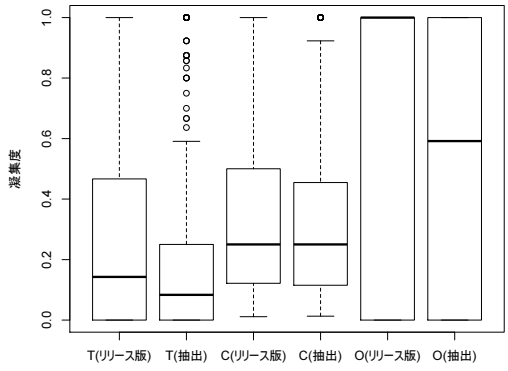
次に，図 3 から，各凝集度メトリクスの結果について考察する．Tightness は全てのソフトウェアにおいて，抽出対象と比較用バージョンの間で有意差がみられた．また Tightness の中央値は全て 0.1 以下であり，メソッド抽出対象となったメソッドの多くが Tightness の値が約 0.1 以下であった．このことから，Tightness が 0.1 以下のメソッドを抽出の対象として推薦することが有用であると考えられる．Coverage は Ant と ArgoUML では有意差があったが，jEdit では有意差が見られなかった．Coverage については，対象ソフトウェア間で結果に差が見られたため，今回対象とした以外のソフトウェアに対しても調査を行い，結果を確認する必要があると思われる．Overlap は全てののソフトウェアで分布に有意差がみられた．しかし，Overlap メトリクスは起点になる変数が 1 つしか存在しない場合は必ず値が 1 になり，実際に比較用バージョンに対する結果では，半数以上が Overlap

表 2　比較結果 (jEdit)

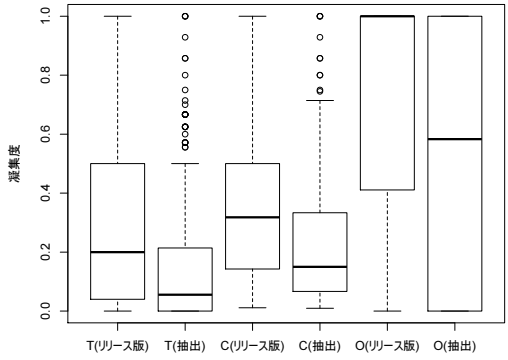
	中央値	p 値
NOS(リリース版)	3	p<0.05
NOS(抽出)	11.5	
Tightness(リリース版)	0.143	p<0.05
Tightness(抽出)	0.083	
Coverage(リリース版)	0.250	0.05<p
Coverage(抽出)	0.250	
Overlap(リリース版)	1.000	p<0.05
Overlap(抽出)	0.592	

表 3　比較結果 (Ant)

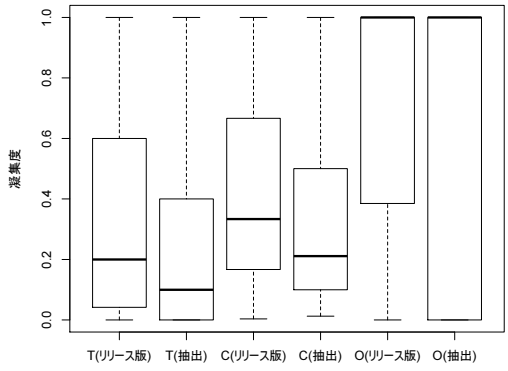
	中央値	p 値
NOS(リリース版)	1	p<0.05
NOS(抽出)	14	
Tightness(リリース版)	0.200	p<0.05
Tightness(抽出)	0.056	
Coverage(リリース版)	0.318	p<0.05
Coverage(抽出)	0.150	
Overlap(リリース版)	1.000	p<0.05
Overlap(抽出)	0.583	



(a) jEdit



(b) Ant



(c) ArgoUML

図 3　メソッドの凝集度分布

の値が 1 になっている．このように，Overlap の値が 1 になるメソッドが多数存在する場合，Overlap を用いてメソッド抽出候補を特定するのは有用ではないと考えられる．

表 4　比較結果 (ArgoUML)

	中央値	p 値
NOS(リリース版)	2	p<0.05
NOS(抽出)	7	
Tightness(リリース版)	0.200	p<0.05
Tightness(抽出)	0.100	
Coverage(リリース版)	0.334	p<0.05
Coverage(抽出)	0.211	
Overlap(リリース版)	1.000	p<0.05
Overlap(抽出)	1.000	

5　まとめ

本研究では，メソッド抽出対象となるメソッドが持つ特徴の定量的調査を行った．調査においては，オープンソースソフトウェアから収集した，抽出の対象となったメソッドとリリースバージョンにおける全てのメソッドに対して，NOS と凝集度の計測を行い，それらの比較を行った．比較の結果，抽出対象となったメソッドは NOS が多く，凝集度が低いことがわかった．また，それぞれの分布を調べることで，NOS と Tightness がメソッド抽出候補の推薦に有用な尺度であることがわかった．

今後の課題としては，まず，対象ソフトウェアと計測する特徴を追加した大規模な調査を行うことが挙げられる．そして，それらの結果を用いて，抽出対象となるメソッドを推薦する手法を提案することが今後の課題である．

謝辞 本研究は，日本学術振興会科研費基盤 (S)（課題番号 25220003）の助成を得た．

参 考 文 献

[1] Fowler, M.: *Refactoring: Improving the Design of Existing Code*, Addison Wesley, 1999.

[2] Murphy-hill, E. and Black, A. P.: Breaking the barriers to successful refactoring: observations and tools for extract method, *Proc. of ICSE*, 2008, pp. 421–430.

[3] Murphy-Hill, E., Parnin, C., and Black, A. P.: How we refactor, and how we know it, *Proc. of ICSE*, 2009, pp. 287–297.

[4] Ott, L. M. and Thuss, J. J.: Slice Based Metrics for Estimating Cohesion, *Proc. of METRICS*, 1993, pp. 71–81.

[5] Stevens, W. P., Myers, G. J., and Constatine, L. L.: Structured design, *IBM Syst.J*, Vol. 13, No. 2(1974), pp. 115–139.

[6] Tsantalis, N. and Chatzigeorgiou, A.: Identification of extract method refactoring opportunities for the decomposition of methods, *Journal of Systems and Software*, Vol. 84, No. 10(2011), pp. 1757–1782.

[7] Vakilian, M., Chen, N., Negara, S., Rajkumar, B. A., Bailey, B. P., and Johnson, R. E.: Use, disuse, and misuse of automated refactorings, *Proc. of ICSE*, 2012, pp. 233–243.

[8] Weiser, M.: Program slicing, *Proc. of ICSE*, 1981, pp. 439–449.

[9] 藤原賢二, 吉田則裕, 飯田元: 構文情報を付加したリポジトリによるメソッド抽出リファクタリングの検出, 信学技報, Vol. 113, No. 24(2013), pp. 19–24.