The University of Osaka
Institutional Knowledge Archive

Title	類似メソッド集約支援ツールの開発
Author(s)	政井, 智雄; 吉田, 則裕; 松下, 誠 他
Citation	ウインターワークショップ2011・イン・修善寺 論文集. 2011, p. 19-20
Version Type	VoR
URL	https://hdl.handle.net/11094/50639
rights	ここに掲載した著作物の利用に関する注意 本著作物の著作権は情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

類似メソッド集約支援ツールの開発

政井智雄[†1] 吉田則裕[†2]
松下誠[†1] 井上克郎[†1]

ソフトウェア保守のコストを増大させる要因として，コードクローンが指摘されている．類似メソッド対（コードクローンを共有するメソッドの対）間のコードクローンは，“テンプレートメソッドの形成”リファクタリングを行うことで集約できる．本論文では，このリファクタリングを支援する手法，およびその手法を実装したツールを紹介し，今後の課題について議論する．

Developing a Tool for Merging Similar Methods

Tomoo Masai,[†1] Norihiro Yoshida,[†2] Makoto Matsushita[†1]
and Katsuro Inoue[†1]

Code clone is pointed out as one of factors to increase cost for software maintainance. “Form Template Method” refactoring can merge code clones between a pair of similar methods. In this paper, we introduce a technique which supports refactoring, and a tool that we have developed.

1. はじめに

ソフトウェア保守を困難にする要因の１つとしてコードクローンが指摘されている[1)-3)]．コードクローンとは，ソースコード中に含まれる一致もしくは類似したコード片を持つコード片のことである．類似メソッド対（コードクローンを共有するメソッドの対）を集約する方法の１つとして，“テンプレートメソッドの形成”リファクタリング[4)]が挙げられる．このリファクタリングを行う開発者は，(1) 類似メソッド対を各メソッドの固有処理と共通処理に分割し，(2) 共通処理を親クラスのメソッドに実装し，(3) 固有処理を各子クラスのメソッドに委譲するように書き換える．しかし，人手で手順 (1) を行うことは容易ではない．なぜなら，(A) 一方のメソッドの共通処理は，他方のメソッドの共通処理と同一であり，かつ (B) 固有処理として特定された部分はメソッド抽出可能である必要がある．条件 (B) については，通常のメソッド抽出リファクタリングについても同じであるが，あわせて条件 (A) を満たす必要がある点が異なる．開発者は条件 (A)(B) を同時に満たすために，試行錯誤を繰り返しながら固有処理と共通処理の範囲を決めることになる．本論文では，手順 (1) を支援するために提案した，固有処理として切りだすコード片の候補を提示する手法[5)]を紹介し，今後の課題について議論する．

2. 提案手法

文献[5)]で提案した手法は，類似メソッド対間の構文上の差分を検出し，その後差分を含みかつメソッド抽出に適したコード片を固有処理として切りだす候補として検出する．候補の検出では，まず差分と同一の範囲に対してメソッド抽出に適しているか判定し，その後段階的に範囲を拡大させながら，同様の判定を行う．また，固有処理を各子クラスのメソッドとして切りだしたとしても，それらのメソッドを呼び出す親クラスのテンプレートメソッド中の呼び出し文に差異があると，その差異を吸収しながら集約を行う必要があり，集約作業が比較的困難になる．そこで，それらメソッド呼び出し文の差異の有無により候補を分類し，集約が比較的容易な候補と比較的困難な候補に分類した．

3. 実　装

この手法は，開発の過程において利用出来るように，統合開発環境 Eclipse の Java 開発キットのプラグインとして実装している．このプラグインを利用する場合は，リファクタリングの対象とする類似メソッド対

†1 大阪大学
Osaka University
†2 奈良先端科学技術大学院大学
Nara Institute Of Science And Technology

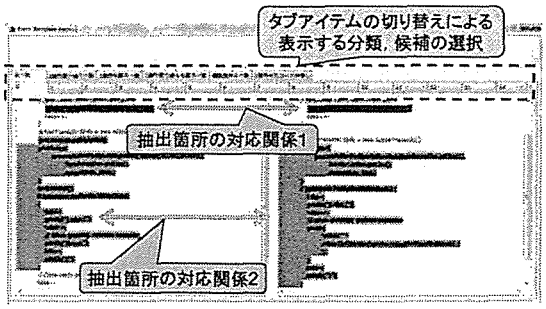

図 1　表示画面
Fig. 1　Snapshot of proposed tool

を決定した上で，Eclipse のメニューからプラグインにより追加された機能を選択する．その後ウィザードの指示に従い，類似メソッド対を選択し，図 1 のようにウィザードに表示される候補を確認した上で，開発者はリファクタリングを行うべきかを判断する．

閲覧する分類および候補はタブアイテムを切り替えることで選択できる．候補の検出や分類の際にいくつかの Eclipse の既存の機能を用いているため，これらの候補を用いたリファクタリングは，Eclipse の既存のリファクタリング機能を利用することで効率的に行うことが可能である．

4. 評価実験

評価実験では，オープンソースソフトウェア ANTLR に含まれている 5 つの類似メソッド対に対して，本手法の適用を行った．検出された候補の一部に対して実際にリファクタリングを行ったところ，外部的振る舞いが変化しないことを確認でき，候補の有用性を確認できた．また，候補の検出数については，ある類似メソッド対については 102,144 個の候補が検出された，その全てがメソッド呼び出し文が一致しない集約作業が比較的困難な候補であった．他の 4 つの類似メソッド対については 9〜695 個の候補が検出され，うちメソッド呼び出し文が一致する集約作業が比較的容易な候補は，9〜150 個であった．よって，それら 4 つのメソッド対については，現実的に閲覧が可能な数の有用な候補が検出できたと考えられる．

5. 今後の課題

今後の課題としては以下の内容が挙げられる．

(1) 一部の類似メソッド対から検出される候補の数は，現実的に閲覧が不可能であった．したがって，検出された候補の中から有意性の高い候補を，閲覧可能と考えられる数だけに厳選するなどの対策が必要である．厳選基準として考えられるものは，抽出するコード片や抽出後の各メソッドから算出できる凝集度の値などがある．また，検出数そのものを減らすために，検出前に基準を設定することで検出する候補を制限する方法も考えられる．

(2) 一部の類似メソッド対からは，集約作業が比較的容易な候補を検出できなかった．この問題を解決するためには，抽出範囲の拡大とは異なる新たな候補の検出方法を考案する必要がある．ステートメントの入替えを行うなどの方法が考えられるが，候補数の増加が懸念される．

(3) 現時点では提示された候補の有用性及び分類の有用性を確かめる実験のみを行っている．したがって，提示された候補が支援として有効であるか確かめる実験が必要である．例えば，被験者に実際に“テンプレートメソッドの形成”リファクタリングを行ってもらい，ツールを用いる場合と用いない場合において作業効率や品質にどのような差が生じるかを調査する，といった方法が考えられる．

(4) 検出される差分や候補によっては，“テンプレートメソッドの形成”リファクタリングよりも，別のリファクタリングが適している場合があるのではないかという意見を得た．他のリファクタングの支援を考慮する場合，類似メソッドの集約に効果的であるリファクタリングについて調査し，その適用可能性を考察する必要がある．

参 考 文 献

1) Jiang, L., Misherghi, G., Su, Z. and Glondu, S.: DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones, *Proc. of ICSE 2007*, Minneapolis, MN ,USA, pp.96–105 (2007).
2) Kamiya, T., Kusumoto, S. and Inoue, K.: CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code, *IEEE Trans. Softw. Eng.*, Vol.28, No.7, pp.654–670 (2002).
3) Komondoor, R. and Horwitz, S.: Using slicing to identify duplication in source code, *Proc. of SAS 2001*, Paris, France, pp.40–56 (2001).
4) Fowler, M.: *Refactoring: Improving the Design of Existing Code*, Addison Wesley (2000).
5) 政井智雄，吉田則裕，　松下誠，井上克郎：テンプレートメソッドの形成に基づく類似メソッドの集約支援，FOSE2010, pp.125–130 (2010).