他クラスのメソッドを利用するコードクローンの調査

2014SE094 鈴木景志
指導教員：横森励士

1　はじめに

ある機能を実現する際に，他のクラスで提供されている機能を利用する場面は数多くみられる．同じ機能を利用する場面が多いほど，メソッド利用などの前後で同じような記述が出現する頻度が高くなり，コードクローン解消を目的としたリファクタリングの対象となりやすいと考えられる．本研究では，Java のソースコードにおいて他クラスのメソッドを利用する記述を含むコードクローンの事例を収集し，それらのコードクローンがどのように解消できるかを調査する．コードクローン解消においてどのような点に着目すべきか，どのようにリファクタリングの基本操作を組み合わせるべきかについて考察を行い，コードクローン解消のためのガイドラインを作成する．

2　クローン解消のためのリファクタリング

リファクタリングとは，ソフトウェアの振る舞いを同一に保ちながら記述を変更することで，内部構造を改善し保守性や理解しやすさを改善する作業である．ファウラーは [2] でリファクタリングにおける基本操作を紹介しており，実作業ではそれらの操作を 1 つずつ適用し，ソフトウェアの中身を再構築する．コードクローンとは，ソースコード中に存在する一致または類似したコード片のことを指す．コードクローンはソースコードを保守する際にコストの増大につながるので，一般的には除去すべきものであることが知られており，肥後らは，コードクローン除去のためのリファクタリング方法を以下のように [1] で紹介している．

「メソッドの抽出」
既存メソッドの一部分を新たなメソッドとして抽出する．コードクローン間に共通するロジックを表すメソッドを 1 つ用意し，コードクローン間の差異部分を引数として渡すことで共通のメソッドで代替する．

「クラスの抽出、親クラスの抽出」
既存クラスの一部分を新たなクラスとして抽出し，既存クラスの機能を新しいクラスに委譲する．抽出したクラスを既存クラスの親クラスとするかは状況に応じて決定する．複数のクラスが多くのコードクローンを共有している場合の既存クラス間のコードクローンの解消に利用する．

「メソッドの引き上げ」
既存のメソッドを親クラスに引き上げる．共通の親クラスを持つ場合は，メソッドの引き上げでコードクローンを解消できる．メソッドの一部分のみが，コードクローンの場合はメソッドの抽出が共通の親クラスが存在しない場合は親クラスの抽出も適用できる．

「テンプレートメソッドの形成」
複数の子クラスのメソッドにおいて，処理の手順は同じであるが詳細な処理内容が異なっている場合に，処理の手順を共通化する．コードクローンを共有しているメソッドの場合は，コードクローンでない部分をメソッドとして抽出し，コードクローン部分を親クラスに引き上げることで解消する．

「メソッドの移動」
メソッドを他のクラスに移動させる．親クラスに移動させる場合は，メソッドの引き上げになる．複数のクラスに存在する重複したメソッドを別のクラスに移動させ，移動させたクラスを利用するように変更することでコードクローンが解消される．

「メソッドのパラメータ化」
メソッド内で利用されている変数やリテラルを引数に変更する．よく似た振る舞いをするものの，異なる値を利用している複数のメソッドがあった場合にコードクローンを削除できる．

3　他クラスのメソッドを利用する記述を含むコードクローンの調査

[1] では，コードクローン解消を目的としたリファクタリングにおける基本操作を紹介し，組み合わせることについて言及しているが，具体的にどう組み合わせるべきかについて言及していない．本研究では，実際のオープンソースプロジェクトを対象に，他クラスのメソッドを利用する記述を含むコードクローンを抽出し，以下の項目について調査する．コードクローン解消においてどのような点に着目すべきか，どう組み合わせるべきか考察する．

項目 1　他クラスのメソッドを利用する部分を含むコードクローンを分類し，どのようなコードクローンが多く見られたかを調査する．図 1 では対象となるコードクローンの分類の例を示している．

項目 2　それらのコードクローンにどのような対処を行えばリファクタリングができるかを調査する．

項目 3　事前もしくは事後に行うべきリファクタリングはどのようなものかを調査する．

調査結果
5 個のオープンソースプロジェクトに対して分析を行った．コードクローンの数は総数 2954 例あり，他クラスのメソッドを利用している記述を含むコードクローンを 246 例抽出した．抽出したコードクローンについて前後のコードとの関連性をもとに分類した．1 つのコードクローンが複数基準を満たす場合にはそれ

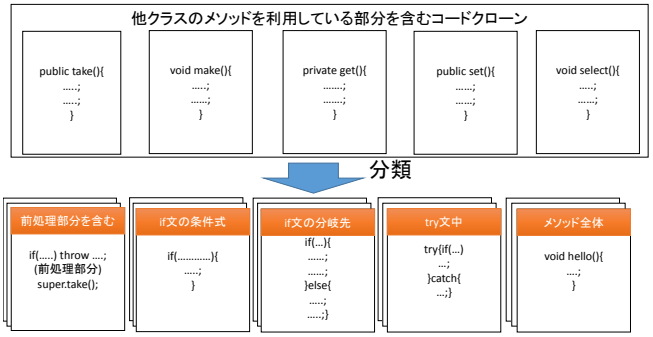

図 1　抽出したコードクローンの分類

ぞれカウントしている．

- 前処理の部分を含めて，他クラスのメソッドを利用している例は 36 例あり，前処理部分が引数や変数の違いがある場合は「テンプレートメソッドの形成」が，完全に一致している場合は「メソッドの引き上げ」が考えられる．事前に必要なリファクタリングとして，「メソッドの移動」でメソッドを適切な場所に移動させる必要がある．例えば，コードクローンが存在するクラスが共通の親クラスを持たない場合に，共通の親クラスを持つクラスにメソッドをまとめるために「メソッドの移動」を行う．
- if 文の条件式の中で，他クラスのメソッドを利用している部分がコードクローンとなっている例は 58 例あり，コードクローンが完全に一致していることが多かったので「メソッドの引き上げ」が考えられる．事前に行うリファクタリングとして「条件記述の分解」で条件が長い場合に if 文を分解する必要がある．例えば，他クラスのメソッドを複数呼び出すことによりメッセージ連鎖が起こり条件式が長くなっている場合「条件記述の分解」を行う．
- if 文の分岐先で，他クラスのメソッドを利用している例は 131 例あり，リファクタリングは then 部で引数や変数の違いが多い場合は「テンプレートメソッドの形成」,「メソッドのパラメータ化」で then 部をリファクタリングし，完全に一致している場合は if 文全体を「メソッドの引き上げ」をすることが考えられる．事前に必要なリファクタリングとして，「メソッドの移動」,「メソッドの抽出」,「条件記述の分解」でクラスや if 文を小さくし，適切な場所へ移動させる必要がある．例えば，if 文の then 部が大きい場合は else 部も大きくなることが多いため「条件記述の分解」でそれぞれに分解する必要がある．
- 他クラスのメソッドの利用を含むメソッドの全体がコードクローンとなっている例は 62 例あり，引数や変数の違いが見られるので「テンプレートメソッドの形成」,「メソッドのパラメータ化」が考えられる．事前に必要なリファクタリングとして，「メソッドの移動」で適切な場所に移動させる必要がある．例えば，異なるオブジェクトに対しメソッドを適用するコードクローンは他クラスで完全に一致していることがあるため「メソッドの移動」で無くす必要がある．
- try 文中で実行する文がコードクローンになっている例は 12 例あり，クラスの一部分がコードクローンであるパターンが多かったため「メソッドの抽出」と「メソッドの引き上げ」が考えられる．事前に行うリファクタリングとして，「条件記述の分解」を用いて小さくする必要がある．例えば，try 文の中に if 文が使われている場合は，try 文全体が巨大になりやすいため「条件記述の分解」で小さくする．

4　考察

調査結果からは，他クラスのメソッドを利用している記述を含むコードクローンは，大きく 5 つの状況に分けることができた．事前または事後にリファクタリングを組み合わせる必要がある場合があると考えられる事例は，80 例抽出することができ，約 30 %のコードクローンで事前に対応が必要であった．どのような点に着目すべきかについては，「コードクローンを有するクラスが共通の親クラスを持つか」や「条件記述や分岐先のブロックが長くなりすぎないか」,「メソッド全体がコードクローンであるか」などを考慮する必要があった．

コードクローンを持つクラスが共通の親クラスを持つ場合はクラスの配置によって,「メソッドの移動」,「メソッドの引き上げ」を使い分ける必要がある．条件記述や分岐先のブロックが長い場合には「条件記述の分解」を組み合わせる必要がある．メソッド全体がコードクローンである場合は，異なるオブジェクトを呼び出している場合に他の記述に拘わらず,「メソッドの移動」を組み合わせる必要がある．という対策が考えられ，これらをもとにガイドラインを作成することで，他クラスのメソッドを利用する記述を含むコードクローンの適切な解消を支援できる．

5　まとめ

本研究では，実際のオープンソースプロジェクトのソースコードに対して調査を行い，他クラスのメソッドを利用することでコードクローンをどのように解消できるか，解消する際に考慮すべきことはあるかについて調査を行った．リファクタリングの際に事前にリファクタリングを組み合わせる必要がある事例を確認し，対応方法を検討した．

参考文献

[1] 肥後芳樹，吉田則裕:“コードクローンを対象としたリファクタリング”, コンピュータソフトウェア，Vol.28, No.4，2011.

[2] マーチン・ファウラー :“リファクタリング”，ピアソン・エデュケーション，2000.