メソッドのインライン展開による オブジェクトの責務単位でのコードスニペットの抽出

大 塲 光 明[†]　大須賀 俊憲[†]
小 林 隆 志[†]　山本 晋一郎[††]

現在我々はプライベートメソッドをインライン展開し，パブリックなインターフェースを通して実行される処理内容を単位とすることで，オブジェクトに設定される責務単位でのコードスニペットの抽出を目指している．このポジションペーパーでは，再利用性の高いコード断片（コードスニペット）を抽出するために必要なソースコードの類似性に関して議論する．

1. はじめに

アンケートには大塲が回答した．

大塲は大学の卒業研究として，大須賀が提案した再利用可能なソースコード片の抽出手法の改良を試みている．それ以前にソースコードの類似性について研究などをしたことはない．また，Java でのプログラミングは大学での講義で初めて学んで以来，2 年間の経験がある．現在は一通り読み書きすることができる．

2. アンケート回答

アンケートへの回答を表 1 に示す．また，一部の回答についてその理由を以下に示す．

2.5 ソース No.5 についての回答理由

フィールド `ll` のクラスが異なるが，これらは実装が異なるだけで，どちらも整数の可変長リストのクラスである．よって，これらのソースが各実装の固有の機能に触れていなければ，いずれの観点においても類似していると考えてよいと判断した．

2.8 ソース No.8 についての回答理由

`sort` メソッドを異なるアルゴリズムで実装しているので，双方で同様のバグが潜む可能性は少ないと考えられる．また，ソート処理のコードを再利用したい場合には，どちらも選ぶことができるだろう．ただ，クイックソートとマージソートには安定性やメモリ使用量に特徴的な違いがあるので，あえて異なるアルゴリズムを採用している可能性もあるとして，観点 A では no と回答した．

表 1 設問への回答

ソース No.	A	B	C	D	X
1	yes	yes	yes	yes	-
2	yes	yes	yes	yes	-
3	yes	yes	yes	yes	-
4	no	yes	no	yes	-
5	yes	yes	yes	yes	-
6	yes	yes	yes	yes	-
7	yes	yes	yes	yes	-
8	no	no	yes	yes	-
9	no	yes	yes	yes	-
10	yes	yes	yes	yes	-
11	no	no	no	no	-

2.9 ソース No.9 についての回答理由

一方では Byte 型，他方では Short 型という異なる型に関する詳細な実装であるから，これらを 1 つにまとめることは不適切であると考えた．ただ，同じ数値に関するクラスなので，その他の観点では類似していると見てよいと思われる．

2.10 ソース No.10 についての回答理由

No.7 と同様に，すべての観点で yes としたが，ただ 1 つ気になるのは，`descriptors[i]` の `getName` メソッドが副作用を持っていた場合，2 つのコードは異なる挙動を示す可能性があるところである．

3. 議　　論

3.1 目　　的

多くのライブラリやフレームワークは大規模化，複雑化しており，その利用方法を調べるには大きなコストがかかる．特に各クラス，各メソッドをどのように組み合わせれば目的の動作を得られるかについては，

† 名古屋大学
Nagoya University
†† 愛知県立大学
Aichi Prefectural University

マニュアルを調べてもわからないことがある．

そこで，ソフトウェア開発者はしばしば，自分や他の人が過去に書いたソースコードから，目的の動作を実現しているサンプルコードを探すことがある．コードスニペットと呼ばれるそのようなコード片を既存のソースコードから自動的に抽出する研究は，XSnippet[1] や MAPO[2] など，多数行われている．

我々もこれまでに，ソースコードの構文木の構造とラベルの類似に基づいてコードスニペットを抽出する手法を考案した[3]．この手法では構文木の構造情報を利用した類似性判定手法[4] を応用し，類似性の高いコード片を効果的に収集し，それらの差異を調べることで，従来の方法では抽出されなかった制御構造などを含んだコードスニペットを抽出する．

現在我々はこの手法を改良し，より多く，より精度の高いコードスニペットを抽出しようと考えている．具体的な改良案として，コード片に含まれるプライベートメソッドの呼び出しをインライン展開するという手法について検討している．

Java でのソフトウェア開発の際，ある処理 A を行うメソッドの中で，処理 A を行う上で必要なある処理 B を行うコードが肥大化してきたとき，ソースコードの可読性や保守性を保つために，処理 B を別のプライベートメソッドに分離することがある．しかし我々のこれまでの手法ではこの点までは考慮していないため，たとえ入力となるソースコード群の中に，アンケートのソース No.7 のように，一方では一部の処理を分離しているが，他方では分離していないという点以外では全く同じコード片があったとしても，それらを類似するコード片とはみなさない．

そこで，パブリックなインターフェースを通して実行される処理内容を抽出単位とし，プライベートメソッドのコードを呼び出し元のコード内で展開するようにすれば，見かけだけでは異なるようにみえるコード片も，オブジェクトに設定される責務単位で見れば類似するものとして抽出することができ，コードスニペットの精度や抽出される量を向上させることができるだろうと考えている．

3.2 課　　題

インライン展開をする際，ただメソッド呼び出しの文を対応するメソッドのコードで上書きするのでは，挿入前のコードと等価にはならない．インライン展開する前と同じ動作となるコードを生成するためには，主に引数とスコープの 2 つについて考える必要がある．

例えばアンケートのコード片 10-2 で，プライベートメソッド `printAttribute` をパブリックメソッド `printAttributes` 内で展開しようとする場合，`printAttribute` のコードで仮引数が使われている部分を，メソッド呼び出しの文で指定している実引数を使うように修正する必要がある．修正の方法として，仮引数の部分をそのまま実引数の記述に置き換えるか，仮引数と同じ名前の変数を，実引数の値を初期値としてコード挿入部分の最初で宣言することが考えられる．前者の方法を適用すると，アンケートのコード片 10-1 とほぼ同じになるが，アンケート回答の方で述べたような副作用の可能性がある．後者の方法を適用すると図 1 のようなコードになるが，同じ処理をしているはずのコード片 10-1 とかなり異なる構造となってしまう．

```
public class StoreAppender {
   public void printAttributes(PrintWriter writer, int indent,
         boolean include, Object bean, StoreDescription desc)
         throws Exception {
      PrintWriter writer_ = writer;
      int indent_ = indent;
      Object bean_ = bean;
      StoreDescription desc_ = desc;
      String attributeName = descriptors[i].getName();
      Object bean2_ = bean2;
      Object value_ = value;

      if (isPrintValue(bean_, bean2_, attributeName, desc_))
         printValue(writer_, indent_, attributeName, value_);
   }
   //その他の処理（約350行）は同じであるため省略.
}
```

図 1　printAttribute の展開例

また，プライベートメソッドの側で変数が新たに宣言されている場合，挿入するコードのスコープを考慮する必要がある．挿入するコードをその前後のコードと同じスコープに置くと，変数宣言の競合が起きる可能性がある．挿入するコードを 1 つのブロックに包むことでこれを回避することができるが，構文木の構造に影響が生じる．

参　考　文　献

1) N.Sahavechaphan et al., XSnippet: mining For sample code, OOPSLA '06, pp.413-430, 2006.
2) T.Xie et al., MAPO: mining API usages from open source repositories, MSR'06, pp.54-57, 2006.
3) 大須賀俊憲, 小林隆志, 山本晋一郎, 阿草清滋: 構文木の構造とラベルの類似性に基くコードスニペット抽出 ウインターワークショップ 2009・イン・宮崎論文集 pp.25-26, 2009.
4) L.Jiang et al., DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones, ICSE '07, pp.96-105, 2007.