2M-4

細粒度なコード再利用のための言語機構メソッド内メソッドの定量的な評価に向けて

小川秀典* 佐藤芳樹† 千葉滋‡

*‡東京大学情報理工学系研究科 †東京大学情報基盤センター

1 はじめに

我々は論文 [1] で，メソッド内に定義でき，細粒度でのコード再利用を可能にするメソッド内メソッドを Java 言語向けに提案した．メソッド内メソッドの有用性の定性的な評価は論文 [1] で示されているが，定量的な評価は文献 [2] のプログラムの構造を考慮せずに実施した不十分なものしかない．

本論文は，メソッド内メソッドの有用性を定量評価するための準備として我々がおこなった調査の結果を示す．メソッド内メソッドは，コード・ブロックをメソッドとして抽出する際に有用だが，そのようなコード・ブロックがどのくらい存在するかを 5 つのオープンソースソフトウェアについて調査した．またメソッド内メソッドは上書き可能であるので Template Method パターンに適用するとコードの簡素化に有効な場合がある．我々は 5 つのオープンソースソフトウェアの中に，そのようなパターンが含まれるかも調査したのでその結果も示す．

2 メソッド内メソッド

メソッド内メソッド [1] (以下，内部メソッド) は，外側のメソッド (以下，内包メソッド) のスコープを継承し，内包メソッドで宣言された局所変数を読み書きできるメソッドである．また，定義クラスのサブクラスで上書きすることもできる．内部メソッドは，メソッド内のコードの一部を独立したメソッドとして抽出し，可読性や再利用性を改善する際に役立つ．図 1 では，select メソッドの内の一部のコードが内部メソッド filter として抽出されている．filter は，学生リストの各要素のうち条件に合致するものを選んで処理を適用する処理を表す．

```
class StudentSelector {
  void select() {
    List<Student> studendts = ...;
    public List<Student> passed = ...;
    public double highestScore = 0.0;
    public double lowestScore = 100.0;
    for (public Student s: students) {
      void filter() {
        if (s.gradYear == 2011) {
          if (s.score > 70)
            passed.add(s);
          if (s.score > highestScore)
            highestScore = s.score;
          if (s.score < lowestScore)
            lowestScore = s.score;
        }
      }
      filter();
    }}}
```

図 1: メソッド内メソッド filter

```
class SubStudentSelector extends StudentSelector {
  void select().filter() {
    if (s.attendance > 0.9) {
      if (s.score > 60)
        passed.add(s);
      if (s.score > highestScore)
        highestScore = s.score;
      if (s.score < lowestScore)
        lowestScore = s.score;
    }}}
```

図 2: select の内部メソッド filter の上書き

内部メソッドはサブクラスで上書きできる．図 2 は，サブクラス SubStudentSelector で filter を上書きする様子を示す．上書きするメソッドは，上書きされる内部メソッドの内包メソッドの局所変数のうち，public 宣言されたものを読み書きできる．

内部メソッドは内包メソッドで宣言された局所変数を読み書きができるため，内包メソッドの多数の局所変数を読み書きしているブロックをメソッドとして抽出するのに便利である．そのようなブロックを通常のメソッドとして抽出すると冗長なコードになりがちである．例えば，メソッドが多数の引数をとったり，複数の戻り値を共有変数を介したり，オブジェクトや配列の形にして返すコードになる．

Toward evaluation of the fine-grained reusability of inner methods

*Hidenori Ogawa, The University of Tokyo
†Yoshiki Sato, The University of Tokyo
‡Shigeru Chiba, The University of Tokyo

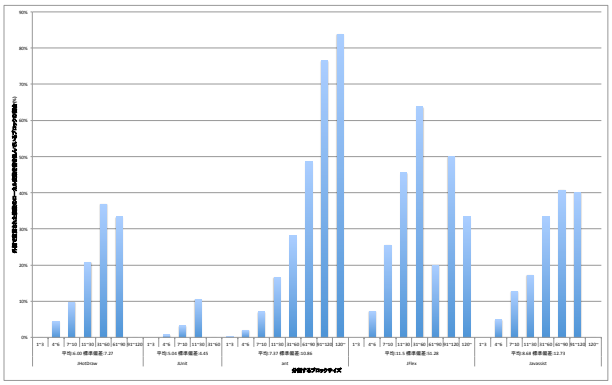

図 3: メソッド内メソッドで抽出するのに適したブロックの割合

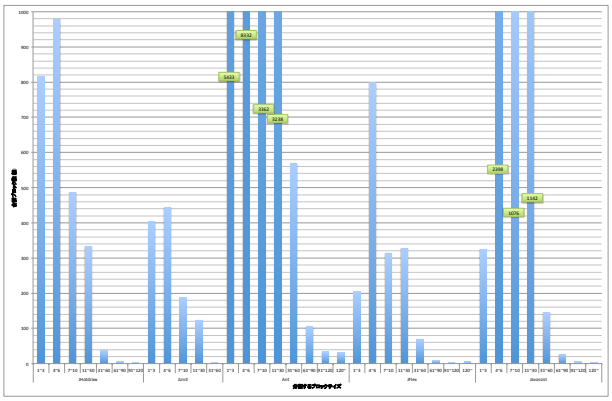

図 4: 大きさ別に分類した調査対象のブロックの総数

3 有用性の調査

メソッド抽出での有用性

内部メソッドが特に有用なのは，外側のスコープで宣言された複数の局所変数に書き込みを実行するブロックを，メソッドとして抽出する場合である．我々は既存の Java コンパイラ JastAddJ を改造し，そのようなブロックの個数を数えた．調査したのは 5 つのオープンソースソフトウェアのプログラムである．評価対象として JHotDraw 5.2，JUnit 4，Apache ant 1.9.2，JFlex1.4.3，Javassist 3 の 5 つを選んだ．それぞれ，Java の描画フレームワーク，Java 用のユニットテスト自動化フレームワーク，Java 用のビルドソフトウェア，Java 用の字句解析器生成ソフトウェア，Java バイトコード変換ライブラリである．

結果を図 3 に示す．我々は各ソフトウェアについて，まずブロックを大きさ (行数) ごと分類し，それぞれの分類について，該当するブロックの個数の割合を調査した．各分類に含まれるブロックの個数は図 4 に示す．ブロックの大きさが大きくなると，内部メソッドによる抽出に適したブロックの候補が多数含まれることがわかる．大きなブロックは抽出の対象になりやすいので，内部メソッドは一定の有用性があるといえる．

Template Method パターンでの有用性

内部メソッドはサブクラスで上書きが可能である．Template Method パターンの primitive operation の実装で内部メソッドが有効な例があれば，内部メソッドの上書き機能の有用性を示せる．そこで我々は改造した Java コンパイラを用いて，前述の 5 つのオープンソースソフトウェアに含まれる Template Method パターンを調査した．

我々の改造コンパイラは，あるメソッドの呼び出しが上位クラスにあれば，そのメソッドを template method 内の primitive operation のオーバーライドと判別する．一般的な template method パターンは，抽象メソッドの呼び出しを指すが，内部メソッドの対象はその限りで無いため，判定条件を緩和した．出力の正しさは [3] の結果と比較して確認した．5 つのソフトウェアについて，改造コンパイラが発見したパターンの出現箇所は，順に 27, 34, 82, 13, 50 個であった．残念ながら，これらの出現箇所に内部メソッドが有効な例は見つからなかった．抽出に内部メソッドが必要なブロックは，抽出されずに Template Method パターンも用いられておらず，類似なメソッドを含んだ冗長なコードのままになっていると考えられる．この点についてさらなる調査が必要である．

参考文献

[1] 平松俊樹, 佐藤芳樹, 千葉滋, “実行速度を考慮した実装法による細粒度でのコード再利用のためのメソッド内メソッド,” 情報処理学会論文誌 プログラミング, vol.6, no.2, pp.45–53, 2013 年 8 月.

[2] 平松俊樹, “実行速度を考慮した実装法による細粒度でのコード再利用のためのメソッド内メソッド,” 修士論文, 東京工業大学 数理・計算科学専攻, 2013 年 3 月.

[3] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, S. T. Halkidis, “Design Pattern Detection Using Similarity Scoring,” IEEE Trans. on Soft. Eng., vol. 32, no. 11, pp. 896–909, Nov., 2006.