メソッド抽出の必要性を評価するソフトウェアメトリックスの提案

三宅　達也[†]　　肥後　芳樹[†]（正員）
井上　克郎[†]（正員）

A Software Metric for Identifying Extract Method Candidates

Tatsuya MIYAKE[†], *Nonmember*, Yoshiki HIGO[†], and Katsuro INOUE[†], *Members*

[†] 大阪大学 大学院情報科学研究科，豊中市
Graduate School of Information and Science Technology, Osaka University, Toyonaka-shi, 560-8531 Japan

あらまし　リファクタリングはソフトウェアの設計品質を改善するための有効な技術である．しかし，どのようなときにリファクタリングが要求されるかを判断するのは容易ではない．そこで，本論文ではメソッド抽出リファクタリングを行う必要のあるメソッドの候補を自動的に識別するためのソフトウェアメトリックスを提案する．また，定性的な評価や適用事例を通して，提案メトリックスを用いることにより，既存のメトリックスでは識別できないメソッド抽出リファクタリングの対象を識別できることを示した．

キーワード　リファクタリング，ソフトウェアメトリックス

1. はじめに

リファクタリングとはソフトウェアの外部的振る舞いを保ったまま，ソフトウェアの内部構造を改善することによりソフトウェアの設計品質を高める技術である [2]．しかし，どのようなときにリファクタリングが要求されるかについての厳密な基準は存在しない．Fowler は過去の経験や知識をもとにリファクタリングが要求される可能性を示すいくつかの兆候 (不吉な匂い) を定義している [2]．また，それらの不吉な匂いのいくつかはモジュールの行数やサイクロマチック数などの複雑度メトリックスを用いて定量的に示すことができる．しかし，本来モジュールは行数や複雑さではなく，機能に基づいて分割することが望ましい．

モジュールの機能はモジュール内の構成要素が協調しあって実現される．本論文ではメソッドの構成要素の協調度合に着目することにより，メソッド抽出リファクタリングを必要とするメソッドの候補を自動的に識別するための新しいメトリックスを提案する．また適用事例を用いて既存のメトリックスとの比較を行い，提案メトリックスの有効性に関して議論する．

2. メソッド抽出を必要とする兆候

典型的なリファクタリング作業の 1 つにメソッド抽出がある [2]．メソッド抽出とはメソッド内のコードの一部を新規メソッドとして抽出するリファクタリングである．以下では，メソッド抽出の必要性を示す既存の不吉な匂いと，それを定量的に表すソフトウェアメトリックスに関して述べる．

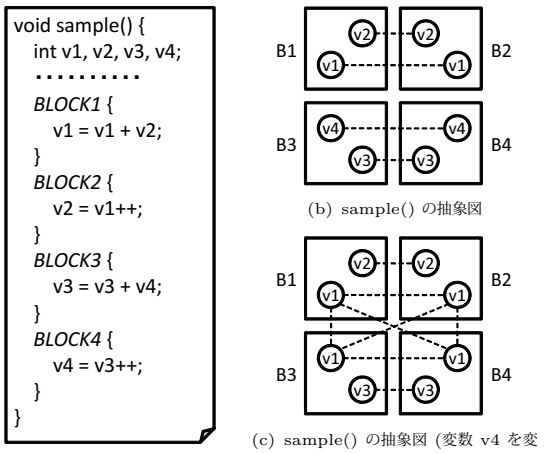
長すぎるメソッド　長すぎるメソッドはメソッド抽出を行うことにより，短くすることが望ましい [2]．メソッドを短くすることにより，可読性や再利用性が向上する．このため，メソッドの行数を表すソフトウェアメトリックスである LOC(Lines Of Code) が大きいメソッドはメソッド抽出の候補であるといえる．

制御フローの複雑なメソッド　制御フローの複雑なメソッドもメソッド抽出の候補である．MaCabe は制御フローの複雑さを表すソフトウェアメトリックスとしてサイクロマチック数 (以降，CYC) を提案している [5]．この値は直観的にはソースコード上の分岐の数に 1 を加えた数を表し，値が大きいほどそのメソッドの制御フローは複雑であるといえる．経験的に CYC は 10 以下に抑えることが望ましいといわれている [5]．

凝集度の低いメソッド　凝集度はモジュール内の構成要素が特定の機能を実現するため協調している度合を表す．メソッドの凝集度を表すメトリックスにはプログラムスライスを用いる手法が提案されている [6]．このメトリックスが低いメソッドはプログラムスライスを用いたメソッド抽出の候補である [4]．凝集度は 7 つのレベルに分けることができ，通常，メソッドの凝集度は最も高いレベルであることが望ましい [7]．プログラムスライスを用いたメトリックスはそのうちの下位 4 つを凝集度の低いモジュール (メソッド抽出の候補) として識別できるが，最上位 (機能的凝集) より低い 2 つのレベル (逐次的・通信的凝集) にあるモジュールを識別することができない場合が多い．これは出力変数 (返り値やメソッド内で変更される属性など) のみに着目してメトリックスが計測されるためである．

3. 提案メトリックス: COB

オブジェクト指向には，処理 (機能要素) および処理に必要なデータ (データ要素) を 1 つにまとめるという重要な考え方がある．同様に，機能要素が必要としないデータ要素は互いに分離しておくことが望ましい．この考えに基づき構成されたモジュールは機能要素とデータ要素が互いに協調しており，凝集度が高くなる．クラスの場合，その度合を表すメトリックスと



(a) sample() メソッド　(b) sample() の抽象図　(c) sample() の抽象図 (変数 v4 を変数 v1 に置換)

図 1　サンプルコード
Fig. 1　A sample code

して LCOM が提案されている [1], [3]．LCOM は属性をデータ要素，メソッドを機能要素とみなし，その協調度合を表す．

本論文では，メソッドの構成要素の協調度合を示すため，メソッド内で使用されている変数をデータ要素，コードブロックを機能要素とみなし，ソフトウェアメトリックス COB(Cohesion Of Blocks) を次のように定義する．b はメソッド内のコードブロックの数，v はメソッド内で使用されている変数の数，V_j はメソッド内で使用されている j 番目の変数，$\mu(V_j)$ は変数 V_j を使用しているコードブロックの数を示す．

$$COB = \frac{1}{b}\frac{1}{v}\sum_{j}^{v}\mu(V_j)\ (0 \leqq COB \leqq 1)$$

COB は LCOM* [3] と同様に，特定のデータ要素と協調する機能要素の割合の平均に着目している．

図 1(a) の sample メソッドコードを用いて COB 計測の例を示す．sample メソッド内の BLOCK1-4 はコードブロックを示す．また，図 1(b) は sample メソッドの構成要素であるコードブロックと使用されている変数の協調関係を抽象化した図である．図 1(b) において四角はコードブロックを，四角内の円はそのコードブロック内で使用されている変数を，点線は 2 つのコードブロックが変数を介して協調していることを意味する．図 1(b) からわかるように，sample メソッドは BLOCK1 と BLOCK2 は変数 v1,v2 を介して協調している．同様に BLOCK3 と BLOCK4 も協調している．一方，BLOCK1，2 で構成されるコード片と BLOCK3，4 で構成されるコード片は協調していない．このとき COB の値は 0.5 となる．変数 v4 の使用を変数 v1 の使用に置き換えたとき，図 1(c) のように全てのコードブロックが変数 v1 を介して協調するため，COB の値は 0.66 に上昇する．さらに，変数 v3 と使用を変数 v2 の使用に置き換えると全変数が全コードブロックで使用され COB は最大値 1 となる．

ここで，呼び出し関係にある 2 つのメソッドについて考える．一方のメソッド内のローカル変数は基本的にもう一方のメソッドからはアクセスできない．引数を利用してアクセスすることは可能であるが，2 つのメソッドが適切に切り分けられているとき引数の数は少なくなる傾向がある．このため，適切に切り分けられた 2 つのメソッドを 1 つにまとめたとき，メソッド内の要素の協調状況は図 1(b) と似たものになり，COB は低くなる．このことから，COB の低いメソッドはメソッド抽出を行い，2 つに分割するべきメソッドの候補であるといえる．

既存のメトリックスと比べ，COB は次の利点を持つ．

- メソッド内の要素の協調度合を表すため，LOC や CYC より機能に基づいた評価が行える．
- LOC や CYC と異なり適切にメソッド抽出した場合のほうが，元のメソッドと新規メソッドのメトリックス値が高くなる．例えば，図 1(b) において BLOCK1 と BLOCK3 をメソッド抽出しても COB は変化しないが，BLOCK1 と BLOCK2 をメソッド抽出した場合は上昇する．
- 出力変数だけでなく全ての変数に着目するため，逐次的・通信的凝集が機能的凝集よりもメトリックス値が低くなる傾向がある．

4. 適 用 事 例

既存のソフトウェア (apache-ant-1.7.0) に対して COB，LOC(空白・コメントを含む)，CYC を計測し，それぞれのメトリックスごとに，メトリックス値の大きさでソートし，上位 20 メソッドをメソッド抽出の候補として検出した．

計測の結果，COB でソートした場合は 0.066 以下，LOC の場合は 146 以上，CYC の場合は 22 以上のメトリックス値をもつメソッドがメソッド抽出の候補として検出された．検出されたメソッドのうち COB でソートした場合のみ検出された 6 つのメソッドを表 1 に示す．6 つのメソッドをメソッド名やコードコメントをもとに調査した結果，*JUnitTestRunner≠main* メ

ソッド以外のメソッドはメソッド抽出の対象として適当であると判断できた．*JUnitTestRunner#main* メソッドに関してはコードコメントが存在しなかったため評価を控えた．

図 2 にメソッド抽出候補の例としてメソッド *Concat#validate* の概略を示す．このメソッドは 6-37 行の範囲に含まれるコードブロック群内では 10 個の変数が，38-89 行の範囲に含まれるコードブロック群では 14 個の変数が使用されていたが，6-37 行と 38-89 行の両方の範囲内で使用されている変数は 2 個であった．つまり，片方の範囲で使用されている変数のほとんどがもう片方では使用されておらず，互いに協調していなかった．このため，COB が減少し，メソッド抽出の候補として検出された．また，このメソッドは 6 行と 38 行のコードコメントから，6-37 行のコード片はバイナリデータとして入力の正当性検証を，38-89 行のコード片はソースコードとして入力の正当性検証を行っていることがわかり，38 行を境に 2 つのメソッドに分割することが適切であると判断できた．

5. まとめと今後の課題

本論文では，メソッド抽出の候補を識別するために，メソッド内の構成要素の協調度合を示すソフトウェアメトリックス COB を提案した．適用事例では COB を用いることにより，LOC や CYC などでは識別できないメソッド抽出の候補を識別できることを示した．また，COB はスライスベースの凝集度メトリックスとは異なる要素に着目して協調度合を評価しているため，両方のメトリックスを用いることにより，機能に基づいたメソッド抽出候補の検出を多角的に行うことができると考えられる．

今後の課題としては，より多くのソフトウェアに対して適用することにより，より定量的に有効性を評価する必要がある．

表 1 COB を用いた場合のみメソッド抽出の候補となるメソッド
Table 1 Extract Method candidates identified only by COB

メソッド名	COB	LOC	CYC
DirectoryScanner#scan	0.066	64	12
ModifiedSelector#configure	0.065	115	18
Parallel#spinThreads	0.065	136	17
JUnitTestRunner#main	0.065	106	19
Concat#validate	0.060	89	20
MacroDef#sameOrSiminlar	0.055	55	16

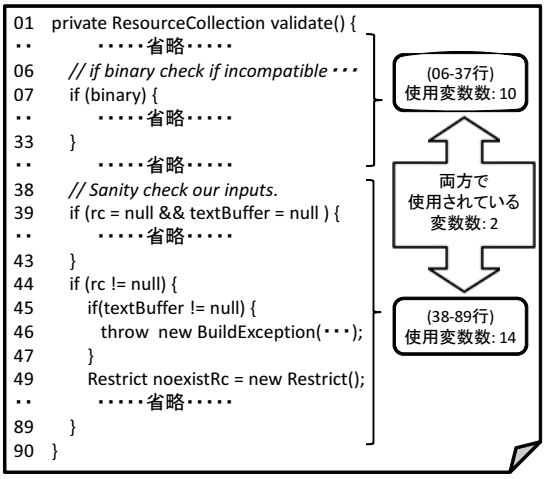


図 2 COB を用いて検出されたメソッド抽出候補の例
Fig. 2 An example of Extract Method candidates identified by COB

謝 辞

本研究は一部，文部科学省「次世代 IT 基盤構築のための研究開発」（研究開発領域名：ソフトウェア構築状況の可視化技術の開発普及）の委託に基づいて行われた．

文 献

[1] S. Chidamber and C. Kemerer. A Metric Suite for Object-Oriented Design. *IEEE Transactions on Software Engineering*, 25(5):476–493, Jun 1994.

[2] M. Fowlor. *Refactoring: improving the design of existing code.* Addison Wesley, 1999.

[3] B. Henderson-Sellors. *Object-Oriented Metrics: Measures of Complexity.* Prentice Hall, 1996.

[4] J. Krinke. Statement-Level Cohesion Metrics and their Visualization. In *Proc. of the 7th International Working Conference on Source Code Analysis and Manipulation*, pp. 37–48, Sep 2007.

[5] T. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, Dec 1976.

[6] L. M. Ott and J. J. Thuss. Slice-based metrics for estimating cohesion. In *Proc. of the 1th International Software Metrics Symposium*, pp. 71–81, 1993.

[7] W. Stevens, G. Myers, and L. Constantine. Structured Design. *IBM Systems Journal*, 13(2):115–139, 1974.

（平成 xx 年 xx 月 xx 日受付）

Abstract Refactoring is an effective technique to improve design quality of software. However, it is not easy to identify where to be refactored. This paper proposes a software metric to automatically identify Extract Method Candidates. A case study shows the proposed metric is useful to identify Extract Method candidates that existing metrics cannot identify.

Key words Refactoring, Software Metrics