B-08

ソースコード解析に基づいた機能群の推定とその分類手法の提案

Identifying and Classifying Functionalities Based on Source Code Analysis

木下　正喬 †　　吉田　則裕 †　　飯田　元 †
Masataka Kinoshita　Norihiro Yoshida　Hajimu Iida

1．　はじめに

ソフトウェア開発プロジェクトの多くは，既存ソフトウェアの保守や拡張を目的としたものである．情報処理推進機構（IPA）ソフトウェア・エンジニアリング・センター（SEC）の調査によると，2008 年度に実施された国内主要ベンダによるソフトウェア開発プロジェクトの内，およそ 40%が既存ソフトウェアを対象としたプロジェクトであった [2]．これらのプロジェクトを進めるためには，開発者は修正の対象となるソースコードを分析し，その機能・構造・振る舞いなどを理解しておかなければならない．

しかし，ソースコード中の個々の記述は詳細すぎるため，全体としてどのように機能しているのかを理解することは難しい．例えば，ソースコードがどのような目的や機能を持っているのか，また機能を追加・変更する場合に修正すべき範囲がどこかなどを知りたい場合には，個々の記述よりもそれらがどのように協調しているのかを広い視点から見る必要がある．このような視点での分析では，設計文書やソースコード中のデザインパターンなどがヒントになるが，修正が重ねられた古いソフトウェアを対象としたプロジェクトなど，これらの情報が利用できないことも多い．

そこで我々は，設計文書が利用できない場合や，ソースコードの構造化が不十分な場合においてもソースコードの効率的な理解を可能にするために，ソースコード中の協調して動作するコード片の集合を「機能候補」として抽出し，それらを「計算ロジック」,「データストア」などの役割別に分類する手法を提案する．

2．　用語の定義

以下では，本稿で用いる用語について述べる．

コード片　コード片とは，ステートメント・ブロック・メソッドなど，ソースコード中の連続した記述のことを言う．以降で説明する最小ブロックや，機能要素もコード片の一種である．コード片は（ファイル ID, 開始行番号，終了行番号，開始桁番号，終了桁番号）の組で表現

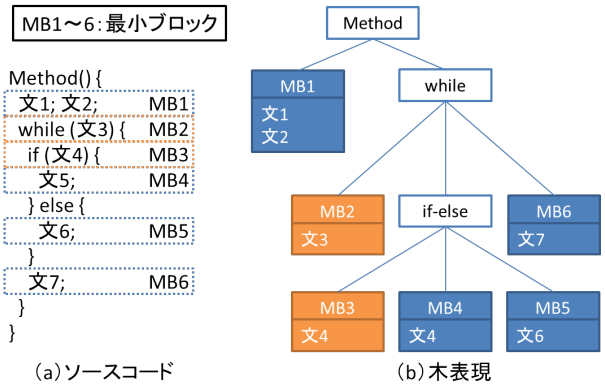

図 1: サンプルコード

することができる．

最小ブロック　最小ブロックとは，プログラミングにおけるブロックが入れ子構造をとるのに対して，入れ子構造を取らないようにさらに細分化したコード片のことである．ブロックや，ブロックとブロックに挟まれた領域のコード片で，さらに内側に他のブロックを持たないようなものを言う．図 1（a）の青とオレンジの枠で囲まれた部分が最小ブロックで，オレンジのものは特に制御構文の条件式を表している．本手法では，1 つの最小ブロックの内部には複数の目的を持つコードが混在することは無いと仮定し，最小ブロックを解析のための最も基本的な単位としている．また，ソースコードは最小ブロックの集合として表現される．

機能要素　機能要素とは，コード片の内その内部に複数の目的を持ったコードが混在していないものを言う．本稿では，機能要素は最小ブロックか，同一の構造に属し特定の機能のために協調して動作する機能要素の集合として再帰的に定義される．例えば，図 1 の最小ブロック MB1～6 は全て機能要素である．また，仮に MB4 と MB5 が協調して動作しているなら，if-else 文全体は機能要素である（詳細は 3 章を参照）．さらに，その if-else 構文と MB6 が協調動作しているなら，while 文全体は機能要素である．最後に，while 文と MB1 が協調動作していなければ，Method は MB1 と while 構文全体の 2 つの機能要素から構成されていることが分かる．

機能候補　大規模なソフトウェアは，数多くの機能の組

† 奈良先端科学技術大学院大学 情報科学研究科．
Graduate School of Information Science, Nara Institute of Science and Technology.

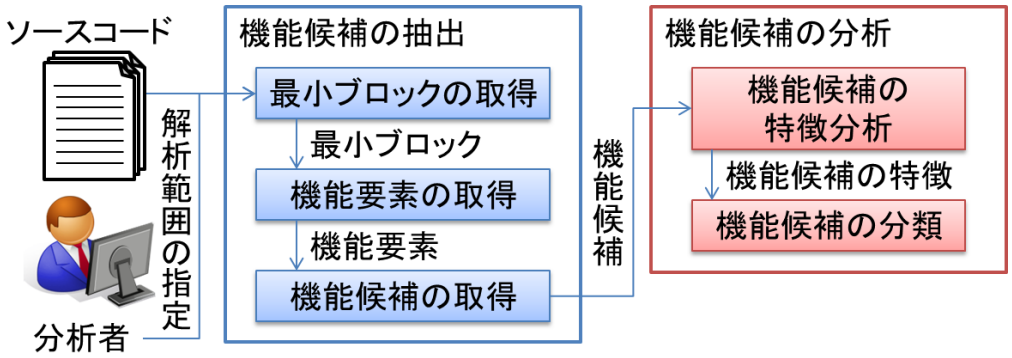

図 2: 提案手法の概要

み合わせによって実現されている．機能候補は，それぞれの特定の機能を実現すると考えられる協調性の高い機能要素の集合を指す．機能候補は機能要素とは異なり，複数のコード片（つまり，不連続部分を含む記述）から構成されることがある．機能候補は，ソースコードのどの範囲がどの機能に対応しているのかを知るための手掛かりとなる．

凝集度　凝集度とは，モジュール内のソースコードの各構成要素が，特定の機能を実現するためにどの程度協調して動作しているのかを表す度合いである．凝集度は，モジュールが単一の機能を実現していれば高い値に，逆に複数の機能を実現していれば低い値になる．

LCOM（Lack Of Cohesion in Methods）はオブジェクト指向設計の複雑度に関する 6 つの CK メトリクスの一つで，クラスの凝集度を表す [1]．LCOM は，特定のデータ要素をいくつの機能要素が使用するかに着目し，クラスのメンバ変数をデータ要素，メソッドを機能要素とみなして計算される．また，COB（Cohesion Of Blocks）はメソッドの構成要素の協調度合いを示すため，メソッドで使用されている変数をデータ要素，メソッド内のブロックを機能要素として LCOM と同様の着眼点から定義されたメトリクスである [3]．

COB は，b をメソッド内のブロック数，v をメソッド内で使用されている変数の数，V_j をメソッド内で使用されている j 番目の変数，$\mu(V_j)$ を変数 V_j を使用しているブロック数として以下のように定義される．

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

提案手法では，凝集度を特定のモジュールを評価するためではなく，凝集度が高くなるような機能要素の集合を機能候補として抽出するために用いる．そのため，LCOM や COB とは異なり任意の機能要素の集合に対して凝集度を求める必要がある．そこで，そのようなメトリクスとして COCP（Cohesion of Code Portion）を定義する．COCP は，機能要素をブロックの代わりに本章で定義したものとし，凝集度を求める対象を任意の機能要素の集合として COB と同様に定義する．

3．　提案手法

図 2 に提案手法の概要を示す．本手法は，機能候補の抽出と得られた機能候補の分析からなる．本章では，それぞれの手法について説明する．

3.1　機能候補の抽出

機能候補の抽出は，以下の手順で行う．

3.1.1　解析範囲の決定

ソースコードの解析範囲を決定するため，自動あるいは手動でソースコード中からメソッドを一つ選択する．メソッドを自動選択では，プログラムのエントリポイントとなるメソッドが選択される．エントリメソッドが存在しない場合や，解析する範囲を絞り込みたい場合には手動での選択を行う．次に，選択したメソッドを外部 API 以外の全てのメソッド呼び出しをインライン展開してできたソースコードを解析の対象範囲とする．これにより，以降メソッド呼び出しは単純なブロックとして扱う．

3.1.2　最小ブロックの取得

ソースコードを最小ブロック単位に分割する．具体的な手順は次の通り．

（1）ソースコードを前方から走査し，ブロック・制御構文を探し出す．

（2-a）単純なブロックが見つかった場合には，ソースコードをブロックの範囲とブロックの前後で計 3 つに分

割する．
（2-b）if 文や while 文などの制御構文が見つかった場合には，ソースコードを制御文，主文及び構文全体の前後に分割する．switch 文や，for 文はそれぞれ等価な if 文，while 文に展開してから同様の操作を行う．
（2-c）ブロックや制御構文が見つからなければそれは最小ブロックである．
（3）（2-a）（2-b）で分解されたソースコードに，（1）～（2）を再帰的に適用する．こうして得られた最小ブロックの集合は，図 1（b）のように解析するメソッドをルートノード，ブロック・制御構文を節ノードとして木構造で表される．

3.1.3　機能要素の取得

次に，ブロックや制御構文などの中で，単一の機能のみを実現していると考えられるものを機能要素とする．前工程で得られた木構造の節（ブロックや制御構文）や葉（最小ブロック）が機能要素になりうる．具体的な手順は次の通り．
（1）木構造の葉ノード（最小ブロック）に対応するコード片を仮の機能要素とする．
（2）木構造の節ノードに関して，条件文以外の全ての子ノードに対して凝集度の提案メトリクス COCP の値を計算する．
（3-a）もし COCP の値が十分に大きいならば（閾値よりも大きいならば），その節ノードに属した子ノードを全て削除し，節ノードを仮の機能要素とする．つまり，その節ノードに対応するコード片は十分に凝集度が高いため，これ以上分割すべきでないと判断する．
（3-b）逆にその値が小さいならば，各子ノードは個別の機能に属している可能性が高いと考え，それらの子ノードに対応するコード片をそれぞれ最終的な機能要素とする．
（4）（2）～（3）を全ての葉ノードが最終的な機能要素となるまで再帰的に実行する．

3.1.4　機能候補の取得

最後に，得られた機能要素を，凝集度を最適化するように組み合わせる．得られた組み合わせの中で，凝集度が閾値以上のものを機能候補として抽出する．組み合わせ最適化の方法には総当り法の他，GA などの高速なメタヒューリスティックの利用を検討している．

3.2　機能候補の特徴別分類

抽出した機能候補を，各種メトリクスや通信の特徴によって役割づけする．それぞれの役割は排他的なものではなく，タグ付けの形で分析者に提示する．役割には，以下のようなものを考えている．

外部インタフェース
外部の API など，ソースコードで提供されていない部分との接続を特徴とする．

内部インタフェース
外部の機能候補から得たデータを，加工せずに他の機能候補へ受け渡すことを特徴とする．

データストア
メンバ変数など，長期にわたって保存されるデータ変数へ高い頻度で直接アクセスを行う．

計算ロジック
メンバ変数など，長期にわたって保存されるデータ変数を加工する．

これらの役割を分類するために，機能候補の内部/外部のデータ変数へのアクセス回数や更新回数，API 呼び出し回数などの，機能候補間の通信に着目したメトリクスを用いる．現在は，機能候補の抽出に取り組んでいるため，役割の分類は今後の課題として取り組む予定である．

４．　まとめと今後の課題

本稿では，ソースコード理解を容易にするための，機能候補抽出手法とその分類手法を提案した．今後は，様々な凝集度やコード片の組み合わせ最適化手法を組み合わせて改良を進めるとともに，実際のソースコードを対象とした評価実験を進めていく．また，機能候補の抽出ができ次第，機能候補の役割分類にも取り組んでいく予定である．

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」の委託に基づいて行われた．

参考文献

[1] S.Chidamber and C.Kemerer:A metric suite for object-oriented design. IEEE Transactions on Software Engineering, Vol.20, No.6, pp.476–493, 1994.

[2] 独立行政法人情報処理推進機構（IPA）, ソフトウェア・エンジニアリング・センター（SEC）:**ソフトウェア開発データ白書** *2009*, 日経 BP 社, 2009.

[3] 三宅達也, 肥後芳樹, 井上克郎: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．電子

情報通信学会論文誌 D, Vol.J92-D, No.7, pp.1071–1073, 2009.