B-009

B Method の仕様記述言語における段階的詳細化を伴う仕様の細分化手法

A Slicing Method for Specifications with Stepwise Refinement Based on Specification Language of B Method

松田 蓮† Ren Matsuda　織田 健† Takeshi Oda

1 はじめに

高品質なソフトウェアの開発手法として形式手法や部品の再利用が注目されている。形式手法の一つである B Method[1] ではソフトウェアが満たすべき仕様を “モデル” として抽象的に記述し、それを実行可能なレベルまで段階的に詳細化した “実装” を作成する。また、Weiser[2] は指定した動作だけを持つ最小限のプログラムをデータフロー等を基に自動的に抽出するアルゴリズムを定義した。これにより、正しさが保証できる範囲での最小限のプログラムを部品として再利用できる。

しかし、B Method のように段階的に詳細化を行うソフトウェアの細分化方法は確立されていない。そこで本研究では Weiser の手法を基に、モデルと実装が連動して記述されるようなソフトウェアの細分化方法を提案する。

2 背景と目的

2.1 プログラムスライシング

Weiser[2] はプログラム中の指定した文の働きに関係する部分を抽出するアルゴリズムを定義した。具体的には、定義したスライス基準を基に、文で変更・参照される変数を求め、プログラム中で関係のある文を抽出する。

2.2 B Method

B Method は集合論と一階述語論理に基づいた形式仕様記述言語による対象のモデル化や仕様の検証などを含むソフトウェア開発手法である [1]。B Method では仕様の抽象的な記述である “モデル” を段階的に詳細化した “実装” を作成し、モデルと実装の組をソフトウェアと見なす。モデルの無矛盾性検証と詳細化の段階間の整合性検証により、ソフトウェアの正しさを保証できる。

モデル・実装間の変数の対応関係は “リンク不変条件” と呼ばれる述語で明示的に記述しなければならない。また、操作による実装変数の状態遷移がモデル変数の状態遷移を満たすかはリンク不変条件を基に検証される。

2.3 B Method のソフトウェアの細分化

中村 [3] は B Method のモデルと実装を整合性を保持したまま細分化する手法の概要（図 1）を示した。本手法により、高信頼ソフトウェア部品（B Method）の作成と再利用が可能になる。また、本手法は “モデル細分化” と “実装抽出” の 2 つの工程で構成され、檜垣 [4] はモデル細分化手法の一部の不備や不足を修正した。

2.3.1 モデル細分化

モデル細分化ではモデルの操作内の相互依存の代入文を細分化の最小単位とし、代入文に関係する変数や制約条件等を抽出し、整合性を保持したまま細分化する。

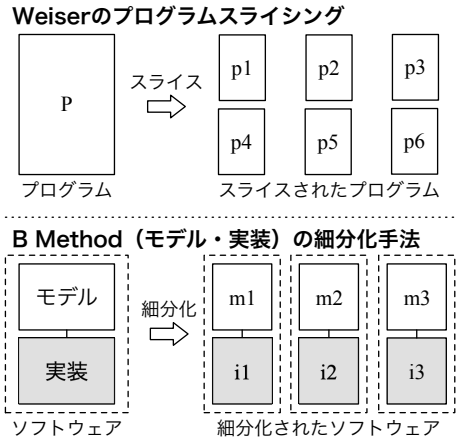

図 1: モデル・実装の細分化の概要

2.3.2 実装抽出

実装抽出ではモデル細分化で生成した細分化モデルに沿うように細分化実装を抽出する。ただし、細分化モデルの記述に沿う形で実装を抽出する工程全体の概要は中村が示したが、詳細な抽出手順は未提案だった。

2.4 研究目的

B Method を対象にしたソフトウェアの細分化はモデル細分化は詳細な手順が示されているが、2.3.2 項で述べた通り実装抽出の手順の詳細は未提案だった。そこで本研究では、Weiser の手法を参考に細分化モデルに沿う形の細分化実装の抽出手法の詳細な定義を目的とする。

3 定義

本稿で用いる用語や式を定義する。なお、紙面の都合で本稿では式や定義の一部を省略する。

3.1 代入文の入れ子構造 PA, CN

B Method で用いる B 言語 [5] では一般的なプログラミング言語の制御構造に当たる式なども代入文と見なされ、そのような代入文は入れ子構造を持つ。任意の代入文 s に対し、PA(s) は s の入れ子構造上の親の代入文を、CN(s) は子の代入文の集合を表す。s が入れ子構造を持つとき、下の階層の代入文も s に含まれるとする。入れ子を持つ代入文は BEGIN, VAR, PRE, ANY, IF, WHILE のみを考える。

3.2 変更変数 D と参照変数 R

代入文 s に対する変更変数 D(s) とは s で値が変更される変数の集合を表し、参照変数 R(s) とは s で値が参照される変数の集合を表す。入れ子の親子の D や R は互いに影響するため、CN を用いた下記の式が成り立つ。

†電気通信大学大学院情報理工学研究科情報学専攻

$$\mathrm{D}(s) = \begin{cases} s \text{ で変更される変数の集合} & (\mathrm{CN}(s) = \emptyset) \\ \bigcup_{s' \in \mathrm{CN}(s)} \mathrm{D}(s') & (\mathrm{CN}(s) \neq \emptyset) \end{cases}$$

$$\mathrm{R}(s) = \begin{cases} s \text{ で参照される変数の集合} & (\mathrm{CN}(s) = \emptyset) \\ V_{ref}(s, Cond(s)) \cup \bigcup_{s' \in \mathrm{CN}(s)} \mathrm{R}(s') & (\mathrm{CN}(s) \neq \emptyset) \end{cases}$$

ここで、$Cond(s)$ は s が属する入れ子の条件式の集合を表し、V_{ref} は代入文 s と $Cond(s)$ に登場する変数の集合を表す。つまり、$\mathrm{R}(s)$ は s と s が属する入れ子の条件式で参照した変数に、子の代入文 s' で参照した変数を加えた集合である。

3.3 モデル変数・実装変数の対応関係

モデル・実装の変数の集合をそれぞれ V_M, V_I とする。LN はモデル変数と実装変数の対応関係を表し、モデル変数 $v_M \in V_M$ が実装変数 $v_{I1}, ..., v_{IN} \in V_I$ に詳細化されているとき、$\mathrm{LN}(v_M) = \{v_{I1}, ..., v_{IN}\}$ である。

4 実装抽出アルゴリズム

4.1 方針

細分化モデルは檜垣 [4] の手法で得られるとする。まず、細分化モデルのモデル変数と実装変数の対応関係 LN をリンク不変条件を基に特定する。次に、細分化モデルの代入文に直接対応する代入文を特定し、変数の依存関係を基に関連する全代入文を芋づる式に特定する。

4.2 具体的なアルゴリズム

本稿では LN の特定手順は割愛する。細分化モデルと実装の入れ子の最も外側の代入文をそれぞれ s_M, s_I とする。

モデルの代入文 s_M に直接関係する実装の代入文の集合 S_{tgt} を $\bigcup_{s_m \in \mathrm{CN}(s_M)} ExtD(s_m, s_I)$ で求める。

$$ExtD(s_m, s_I) = \begin{cases} \left\{ x \;\middle|\; \mathrm{D}(x) \subseteq \bigcup_{v \in \mathrm{D}(s_m)} \mathrm{LN}(v) \right\} & (\mathrm{CN}(s_I) = \emptyset) \\ \bigcup_{s'_I \in \mathrm{CN}(s_I)} ExtD(s_m, s'_I) & (\mathrm{CN}(s_I) \neq \emptyset) \end{cases}$$

得られた S_{tgt} の各代入文 s_{tgt} について、s_{tgt} に間接的に関係する実装の代入文の集合を $ExtR(s_{tgt})$ と $DscR(s_{tgt}, s)$ で求める。なお、代入文 s が WHILE の内側にあるとき $\mathrm{IW}(s) = true$ で、$\mathrm{pred}(s)$ は $\mathrm{CN}(\mathrm{PA}(s))$ の中で s よりも前（構文木では左）にある代入文の集合を表す。

$$ExtR(s_{tgt}) = \begin{cases} (1)\ (\mathrm{PA}(s_{tgt}) \neq \emptyset) \text{ の場合} \\ \quad \begin{cases} (a)\ \mathrm{IW}(s_{tgt}) = true \text{ の場合} \\ \quad \bigcup_{s' \in \mathrm{CN}(\mathrm{PA}(s_{tgt}))} DscR(s_{tgt}, s') \cup ExtR(\mathrm{PA}(s_{tgt})) \\ (b)\ \mathrm{IW}(s_{tgt}) = false \text{ の場合} \\ \quad \bigcup_{s' \in \mathrm{pred}(s_{tgt})} DscR(s_{tgt}, s') \cup ExtR(\mathrm{PA}(s_{tgt})) \end{cases} \\ (2)\ (\mathrm{PA}(s_{tgt}) = \emptyset) \text{ の場合} \\ \quad \emptyset \end{cases}$$

$\mathrm{LN}(xx) = \{xx\_i\}, \mathrm{LN}(yy) = \{yy\_i\}$

細分化モデル

```
op = s_m0
  BEGIN
  s_m1 xx := yy  ||
  s_m2 yy := xx
  END
```

実装

```
op = s_i0
  VAR tmp IN
  s_i1 tmp := xx_i ;
  s_i2 xx_i := yy_i ;
  s_i3 yy_i := tmp ;
  s_i4 zz_i := tmp
  END
```

図 2：細分化モデルと実装の例

$$DscR(s_{tgt}, s) = \begin{cases} \begin{cases} \{s\} \cup ExtR(s) & (\mathrm{D}(s) \subseteq \mathrm{R}(s_{tgt})) \\ \emptyset & (\mathrm{D}(s) \nsubseteq \mathrm{R}(s_{tgt})) \end{cases} & (\mathrm{CN}(s) = \emptyset) \\ \bigcup_{s' \in \mathrm{CN}(s)} DscR(s_{tgt}, s) & (\mathrm{CN}(s) \neq \emptyset) \end{cases}$$

4.3 例

提案手法の具体例（図 2）を考える。

$$\begin{aligned} ExtD(s_{m1}, s_{i0}) &= \bigcup_{s'_I \in \mathrm{CN}(s_{i0})} ExtD(s_{m1}, s'_I) \\ &= \emptyset \cup \{s_{i2}\} \cup \emptyset \cup \emptyset \\ &= \{s_{i2}\} \end{aligned}$$

同様に、$ExtD(s_{m1}, s_{i0}) = \{s_{i3}\}$ なので $ExtD(s_{m0}, s_{i0}) = \{s_{i2}, s_{i3}\}$ である。これに対し、$s_{i0}, ..., s_{i4}$ の各参照変数 R（ここでは右辺に登場する変数）を考慮すると、$ExtR(s_{i2}) = \emptyset$ かつ $ExtR(s_{i3}) = \{s_{i1}\}$ となる。以上の計算により、op の細分化モデル（図 2 左）に必要な実装の代入文 $\{s_{i1}, s_{i2}, s_{i3}\}$ を抽出できる。

5 考察

提案した B Method のソフトウェアの細分化手順はモデルに沿う形で実装をスライスする点が [2] と異なる。

ただし、$ExtR$ は WHILE の内側に相互参照する代入文があると再帰が停止しない為、実装の際は代入文へのラベル付け等の再帰を停止させる工夫が必要である。

6 おわりに

本稿では B Method のソフトウェアの細分化手法を定義した。 今後は細分化ツールの実装と評価実験を行う。

参考文献

[1] 来間 啓伸. B メソッドにおける形式仕様記述. 近代科学社. 2007.

[2] Mark Weiser. Program Slicing. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-10, NO. 4, JULY 1984.

[3] 中村 丈洋. B Method における部品再利用によるソフトウェア合成と高信頼ソフトウェア部品の整備. 電気通信大学 電気通信学研究科 博士（工学）学位論文, 2014.

[4] 檜垣 廉, 織田 健. 演算の可換性と結合性、型に配慮した形式仕様の式表現の統一のための項書換えアルゴリズム. 情報処理学会第 85 回全国大会講演論文集, vol.1 pp.273-274, (2023.03).

[5] ClearSy. B LANGUAGE REFERENCE MANUAL - VERSION 1.8.10.