ドメインモデルの形式的記述手法ならびに要求仕様獲得への活用

6L－3

竹内 寛† 小野 康一‡ 深澤 良彰†

†早稲田大学理工学部 ‡日本アイ・ビー・エム (株) 東京基礎研究所

1 はじめに

ドメイン分析がソフトウェアの再利用性を向上させることはコンセンサスとなりつつある。ドメイン分析とは、同一問題領域におけるソフトウェアシステムを構築するのに用いられる情報を、将来の再利用のために獲得・組織化する工程を指す。この過程で得られる情報を基に、ドメインモデルを構築する。そして、このドメインモデルを軸にソフトウェアの開発を行なうことになる。これは、従来のソフトウェア部品を用いたボトムアップ的な手法に対するアンチテーゼとして生まれてきた、抽象化・具体化を柱とする手法である。

現状の研究では、ドメインモデルは主に図示を基本とし、図中の表記も、自然言語を中心とした非形式的記述を用いるのが主流である。また、言語表記を目的にしていても形式的手法を施しているものは少数である。モデル自体が非形式的なため、その活用プロセスも不明確になってしまっている。記述される情報の正確性や機械による支援を考えた場合、形式的記述方法が適当であるのは言うまでもない。

さらに、現状の研究で散見されるドメインモデルは、問題領域に普遍的に存在する物を表現しているが、それに加えて領域依存の情報の存在を考慮する必要がある。すなわち、普遍的な状態に対して、具体化をするにあたってどのようなオプションが考えられるか、情報を提示して具体化プロセスを支援する必要がある。

このような現状を踏まえ、本研究ではオブジェクト指向概念を取り入れ、抽象度に応じて階層構造を持った形式的なドメインモデル記述言語開発を目指し、それによってモデル活用プロセスの形式化を目指す。なお、支援環境は要求分析工程とする。

2 本研究におけるドメインモデル

本研究では、ドメインモデルを以下の構成要素から成るものと定義する (図１参照)。

- 同一問題領域に存在する複数の類似な要求の重複部分を抽象化したモデル。
- 各要求固有な部位。

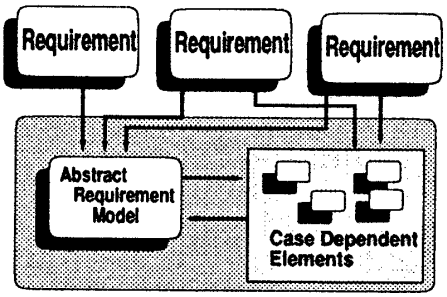

図 1: 本研究におけるドメインモデルの仕組み

入力として事例依存の要求仕様を考える。そこから共通な部分を抽出し、抽象的なモデルを作成する。そこで切り捨てられる情報は、別途保存し、具体化の際に使用する。新規事例の要求分析過程で得られる情報を基に、抽象モデルを具体化して新規案件の要求仕様とする。

3 モデルの形態

抽象化の際に切り捨てられ、具体化の際に付加される情報は、２種類に大別される。すなわち、1.事例によっては問題領域に出現しないオプショナルなデータ・操作・オブジェクトと、2.類似の概念として集約されるデータ・操作・オブジェクトである。この２種類の付随情報と、それを切り落とした結果作成される抽象的なモデル間の相関関係を保持し、統一的なモデルとするためには、オブジェクト指向の枠組みを利用するのが最適である。すなわち、１の場合は主に継承を用いて汎化・特化関係として記述する。２の場合は集約対象の要素をパラメータ化して多相性を用いて記述する。以下、本研究の特徴であるパラメータ化を中心に言語の概要を説明する。例として、図書館の蔵書管理システムを挙げる。

3.1 属性表記

本研究では、オブジェクトの属性は、変数としての一般的なデータと、オブジェクトの状態をあらわすシンボルからなると考える。まずデータの取扱いから説明する。同一問題領域においても、使用されるデータは事

Formal Method in Describing Domain Models and its Application to Requirement Acquisition.

Hiroshi Takeuchi†, Kouichi Ono ‡, Yoshiaki Fukazawa †

† School of Science & Engineering, Waseda University

‡ Tokyo Research Laboratory, IBM Japan

例によって微妙に異なる。本研究では、これを抽象度によって階層的に記述する。概念的には、データ構造は下記の図のようになる。

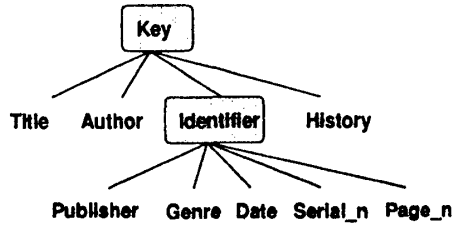

図 2: データ属性の候補

各蔵書に対して key と identifier という 2 つの抽象的な要素が存在し、これが抽象モデルに含まれる。ここで例えば identifier は、Publisher, Genre, Date, Serial Number, Page Number という 5 つの要素を必要に応じて任意に組み合わせることで実体となる。本システムでは、こういった概念を ≪ ... ≫ で要素を囲むことによって表現する。また、パラメータを用いた汎化・特化構造を表す為に、Abst というデータ型を導入し、多段階の階層は以下のような入れ子構造を持ったデータ構造として記述する。

```
Books: book(1 .. MAX) {
  Attribute: Abst:Key
Abst:Key = ≪
        String: Title
        String: Author
          Set: where ∈ { In-Shelf, out } } C
        Abst: Identifier
        Set History: History(1 .. N){
          String: date-for-out
          String: date-for-return
          Object: User() } ≫                  } A

          Abst: Identifier = ≪
              String: Publisher
              String: Genre
              String: Date
              Naturals: Serial-number
              Naturals: Page-number ≫}        } B
```

ここで A と B の差異について意識する必要がある。すなわち A は要素が固定している構造体であり、B は抽象的なパラメータである。新規事例からの入力によって B の候補からいくつかの要素が選択され、具体化されて A のような構造体となる。

次に状態の具体化について触れる。1 つの状態を複数の状態に具体化すると、それぞれに前提条件を付加する必要が生じる。以下に例 (上記 C 参照) を示す。

```
  Lib.book(N).where = In-shelf ⇒
( (Lib.book(N).where = In-opened-shelf)
| (Lib.book(N).where = In-closed-shelf) )
```

- Condition: (Lib.book(N).where = In-opened-shelf) = (Newer(Lib.book(N).identifier.date-of-publication, DATE)) ∨ (Lib.book(N).identifier.genre = GENRE1)

- Condition: (Lib.book(N).where = In-closed-shelf) = (Older(Lib.book(N).identifier.date-of-publication, DATE)) ∨ (Lib.book(N) .identifier.genre = GENRE2)

上記の例では、図書が書架に入っている状態 in-shelf を、開架式の書架と閉架式の書架に具体化している。それぞれについて新たな前提条件が記述されている。この前提条件は、任意の関数が図書の状態を上記の状態 (in-opened-shelf など) へ遷移させるような場合、その関数の前提条件として付加される。

3.2 機能表記

個々の事例における要求に従ってデータを決定すると、同時に機能も具体化される。本研究では、抽象度の最上位に位置する機能を仮想的な関数として定義し、その引数などもパラメータ化されたものを与える。具体的な要求事例や、先行して行なわれるデータの具体化からの入力によって、引数やその型を具体化し、機能の前提終了条件を付加し、機能自体を具体化する。記述例の一部を以下に示す。

```
  /* Declaration of top hierarchy */
  Virtual Method: Lend-Book( Abst: Book.Key )
                  return( Boolean: Result )
  /* Description of abstract function */
  Method: Lend-Book(Abst: Identifier )
Local Var. M
```

- Pre-Condition: {
  Identifier = Lib.book(M). **identifier**
  ∧ Lib.book(M).where = **in-shelf**
  ∧ ≪ Function-user.Verify-qualification() = True ,
  Lib.book(N). **identifier** = **IDENTIFIER** ≫ }

便宜上、具体化される可能性のあるパラメータを下線つき太字で示した。関数の実体は、パラメータを具体化して前提条件の候補を選択することによって生成される。例えば、貸し出しに際して利用者を制限し、また貸し出し図書を開架図書に限定している事例があるとする。その場合は、上記の ≪ ... ≫ 内の一番目の式を選択し、in-shelf を in-opened-shelf に具体化することによって具体事例を表現することが出来る。

4 おわりに

本研究では、今まで非形式的であったドメインモデルへの形式的アプローチを試み、また抽象度に応じた階層的な記述言語を開発した。状態を抽象的なシンボルとして取り扱うことで、要求を抽象的な形態のままある程度形式的に記述することができ、バリエーションの記述性を獲得することができた。今後、記述性や可読性などの評価を行なって行く予定である。