ソフトウェア工学 118−19
（1998. 3. 11）

形式的オブジェクト指向分析モデルFO∀Mの構築法とその支援環境

青木利晃† 古川順一† 片山卓也†

現在オブジェクト指向方法論が提案されており、その典型的なものとしてOMT法がある。しかしその記述や意味の定義が厳密ではない。我々はこれを補う為、形式的オブジェクト指向分析モデル(FO∀M)を提案しているが、定義されているのは静的な関係のみでモデルの構築法までは定義されていない。そこで本論文ではFO∀M概念に基づく分析モデルの構築法を提案し、それを支援する環境の作成例を紹介する。

A Method for Constructing Formal Object-Oriented Analysis Model and Supporting Environment

Toshiaki Aoki,† Junichi Furukawa† and Takuya Katayama†

Recently object-oriented methodologies are focused and OMT is typical of them. But it's syntax and semantics are not defined strictly. We proposed *Fomal Object − oriented Analysis model*(FO∀M) to solve this problem. However, only static relations are defined in FO∀M, so method for constructing models must be defined. In this paper, a method for constructing models based on conceptions of FO∀M and a supporting environment are proposed.

1. はじめに

現在様々なオブジェクト指向方法論が提案されており、大規模なシステム開発に於いても適用され始めている。中でもOMT法[1]は典型的な方法論であり、オブジェクトモデル、動的モデル、機能モデルの3つのモデルを用いてシステムのモデル化を行なう。3つのモデルを用いればシステムの分析を様々な視点から行なえるが、これら3つのモデル間には一貫性が保証されなければならない。しかしOMT法では記述や意味の定義が厳密でないため、3つの図に分散してしまった情報から分析結果の正しさを確認することは非常に困難である。また、形式的な取り扱いが出来ないために、大規模なシステムでは不可欠な計算機による支援もまた難しい。

これに対して、以上の問題を解決するために提案された「形式的オブジェクト指向分析モデル(*Formal model for Object-oriented Analysis Model*：FO∀M[2)~4)]」では3つのモデルに記述の厳密な定義を与えられており、3つのモデルの関連が明確にされている。OMT法よりもより形式的な取り扱いが可能であるため、計算機による支援の幅も広いと考えられる。

しかしFO∀Mは分析モデルの構築方法までは支援していない。そこで本論文では分析モデルを構築する際に必要となる関係や規則などを抽出し、それにそった分析支援環境を提案する。

我々はまず、1つの例題に対してさまざまなアプローチで分析を行った。その結果、FO∀Mに基づいたモデルの構築法の性質として、モデル構成要素の抽出の規則が存在することを発見した。このような規則をポリシーと呼ぶ。ポリシーは、モデルを構築する際の対象ドメイン依存の規範を表現するものであり、効率良くシステムを分析する方法を規定している。そこで、ポリシーがモデル構築メカニズムに対してどのような影響を与えるのかを明らかにするために、まず、FO∀Mにおいて一般的に当てはまるモデル構築メカニズムを形式化し、その枠組みでポリシーがどのように機能しているかを形式化する。このように形式化されたモデル構築メカニズムとポリシーはモデル構築支援法の基礎を与え、又、十分に計算機で取り扱い可能なものと

† 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,Japan Advanced Institute of Science and Technology

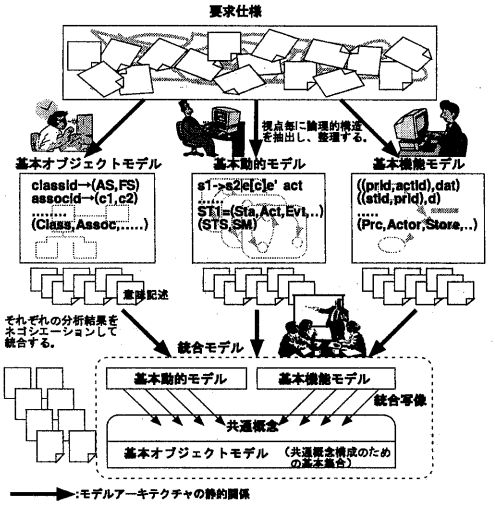

図1　FO∀M の概念

なる。

2. FO∀M

FO∀M では、システムの構成的側面、動的側面、機能的側面を各々閉じた概念の下に形式化し、基本オブジェクトモデル、基本動的モデル、基本機能モデルを構成する。また各々独立に構成された3つの基本モデルは、統合写像によって1つに結びつけられる。

各基本モデルは、その中の要素を識別子集合として表現することで形式化される。例えば基本オブジェクトモデル中に存在するクラスはその識別子集合 **ClassID** に含まれる要素として扱われる。更に統合写像では基本動的モデル及び基本機能モデル中の要素を基本オブジェクトモデル中の要素を用いて表現する。これらの概念を図1に示す。

2.1 基本モデル

基本モデルはOMT 法で用いられている3つのモデルを、各々閉じた概念のみを用いて定義する。対象システムの各側面は基本モデルで定義される要素を用いて分析されるため、抽出すべき成分を明確にして形式化されなければならない。

FO∀M における基本モデルではモデル構築要素に対する意味記述と、それらによって構成されるモデル記述が明確に区別されている。基本モデルにおいて抽出されるのは各側面を構成する要素とそれらの関係である。

例えば基本オブジェクトモデルで定義されるべき内容は、

(1)　システムを構成するオブジェクトを代表するクラス
(2)　クラス間の関係

がある。

クラスは実体化されるオブジェクトの振舞いを実現するために属性／操作を持つ。これらはオブジェクトが所有するべき機能単位を表現したものであり、システム機能の最小単位をオブジェクトへと分割することに繋がる。そこで FO∀M では以下の様な識別子集合を導入している。

- **ClassID** : クラス識別子の集合
- **AttrID** : 属性識別子の集合
- **FuncID** : 関数識別子の集合

基本オブジェクトモデルでは要求仕様記述から必要な機能単位を識別し、オブジェクトの所有物として構造化するのが目的である。関数の機能などは意味記述としてドキュメント化される。

あるクラスが所有する属性集合集合 $AttrSet$ と関数集合 $FuncSet$ は、クラス式として定義されており、$(AttrSet, FuncSet)$ の形で表現される。クラス識別子とこのクラス式は、以下の写像を定義することにより対応付けされる。

$$\mathcal{CM}_{class}[ClassSet] : ClassSet \rightarrow ClassExp$$
$$where$$
$$ClassExp = Pow(\mathbf{AttrID}) \times Pow(\mathbf{FuncID})$$
$$ClassSet \subseteq \mathbf{ClassID}$$

以上で定義したクラスに対してその関係を定義する。例えば継承関係を例にとると、継承関係ではクラス構造の包含関係を定義する。継承関係はクラスを構造により分類出来る様にするだけではなく、同一構造を持つクラスを用いてより詳細な構造を持つクラスの定義を可能にする。

FO∀M における継承関係はその識別子集合を表す **InherID** と、継承関係式 $InherExp$ 及び継承関係識別子から継承関係式への写像 $\mathcal{CM}_{inher}$ を用いて定義される。親クラス $classid$ と子クラス $classid_1, \cdots, classid_n$ 間の継承関係式は

$$(classid, (classid_1, \cdots, classid_n))$$

の様に表現される。またクラス識別子集合 CS 上の継承関係式 $InherExp(CS)$ と写像 $\mathcal{CM}_{inher}[CS]$ は以下の様に定義される。

$$InherExp(CS) = CS \times (CS \times \cdots \times CS).$$
$$\mathcal{CM}_{inher}[CS] : \mathbf{InherID} \rightarrow InherExp(CS).$$
$$where$$
$$CS \subseteq \mathbf{ClassID}$$

2.2 統合写像

以上で定義した各々の基本モデルは、対象システムを直交した3つの視点に完全に分離して分析することを可能にしている。この様にモデル化された対象システムでは、ドキュメント化された意味記述を基に各モデルを対応付けることが出来る。FOVM では統合写像と呼ばれる写像を用いてこれを行なう。

各モデルの視点は直交しているが、対象システムの共通部分をモデル化している部分が存在する。この様な3つの基本モデルに共通した部分の対応関係を考えることにより、各側面の分析結果を反映した統合モデルを構成することが可能になる。以下例として基本動的モデル中のアクション識別子に関する統合写像を考える。

基本動的モデルではオブジェクトの状態と動作のタイミングがモデル化されており、アクションや遷移条件などの意味記述は別にドキュメント化される。この意味記述を基に、基本動的モデルの構成要素を共通概念へ対応付ける。ここで共通概念は、基本オブジェクトモデルで定義されている概念を用いてオブジェクトの振舞いや機能を表現する式を定義する。

アクションの動作に関する意味記述は、基本動的モデル構築の際に別途ドキュメント化されている。この動作を基本オブジェクトモデル中で定義されている属性／操作を用いて構成する。この構成に関して以下の様なアクション式を用いる。

$$aexp = c.f(term,\cdots,term)|c.a := term|$$
$$e.a := term|aexp;aexp$$
$$where$$
$$c \in \mathbf{ClassID}, a \in \mathbf{AttrID},$$
$$f \in \mathbf{FuncID}, e \in \mathbf{EventID}.$$

以上の様なアクション式の集合を $AExp$ とする。

アクション識別子は写像 I_{act} により上記アクション式で構成される動作に対応付けられる。I_{act} は以下の様に定義されていて、これをアクションの統合写像と呼ぶ。

$$I_{act} : \mathbf{ActionID} \rightarrow AExp.$$

3. 段階的構築メカニズムの形式化

大規模システム開発においては、最初から各々の側面における概念をすべて抽出することは不可能であり、中間的なモデルの構築及び分析作業を繰り返し行うことにより最終的な分析モデルを得ることになる。一方、多視点からモデリングを行うような手法において段階的構築を行う場合は、視点間を横断した過程などが複雑にからみあった状況を生じる。そこでまず、FOVM 上でモデルを段階的に構築する際の、構築メカニズムを明らかにして、形式化を行う。

3.1 整合条件

FOVM では、モデルを構成する要素とそれらの間の依存関係が明確に示されているので、このような構築プロセスの策定に明確な指針を与える。基本モデルと統合写像を構成する要素の構成関係により、要素間の依存関係を図2のように抽出することができる。この依存関係を有向グラフによって以下のように定義する。

定義 3.1 構成要素間の依存関係を示す依存グラフ DG を以下のように定義する。

$$DG \stackrel{def}{=} (\mathbf{C},\mathbf{A})$$
$\mathbf{C}$: 図2に出現するすべての構成要素集合
$\mathbf{A}(\subseteq \mathbf{C}\times\mathbf{C})$: 図2に出現する依存関係

■

この依存関係はある時点で構築されているモデルが十分に意味を持ちうる基準として用いることができる。

ここで以下のような記法を用いる。

記法 3.1 中間段階のモデルを添字を用いて表現する。ただし、添字の値はそのモデルが作成された時間的な順序関係を表現する。

Example

$$\mathbf{Class}_1, \mathbf{Class}_2, \cdots\ OM_1, OM_2, \cdots$$

■

DG により表現される依存関係は、モデル中のある要素を構築する際にそれが他のどの様な要素を用いて構成されるかを示している。例えば、属性識別子集合 **AttrID** 中の識別子 $attrid$ はクラス式 $classexp$ を構成するために使用される。つまり図2中の2つの集合 **AttrID**,$ClassExp$ について $(\mathbf{AttrID}, ClassExp) \in \mathbf{A}$ であるが、これは $ClassExp$ が表現する集合の要素が **AttrID** の要素を用いて構成される可能性があることを示しているのみである。

一方、n 段階で構築されたモデルでは、実際にそれらの要素を用いて構成されており、要素間の構成関係は $\mathbf{AttrID}_n \times ClassExp_n$ の要素で表現されることになる。このように実際に構築されたモデル中に出現する要素集合 C_α, C_β 間の構成関係を $R_n(C_\alpha, C_\beta)$ と書くことにする。ここで、n 段階で構築されたモデルの構成要素間の構成関係を定義する。

定義 3.2 n 段階目に構築されたモデルのすべての構成要素集合の集合を $\mathbf{C}_n$ とすると、その時点のモデルの依存グラフ DG_n を以下のように定義する。

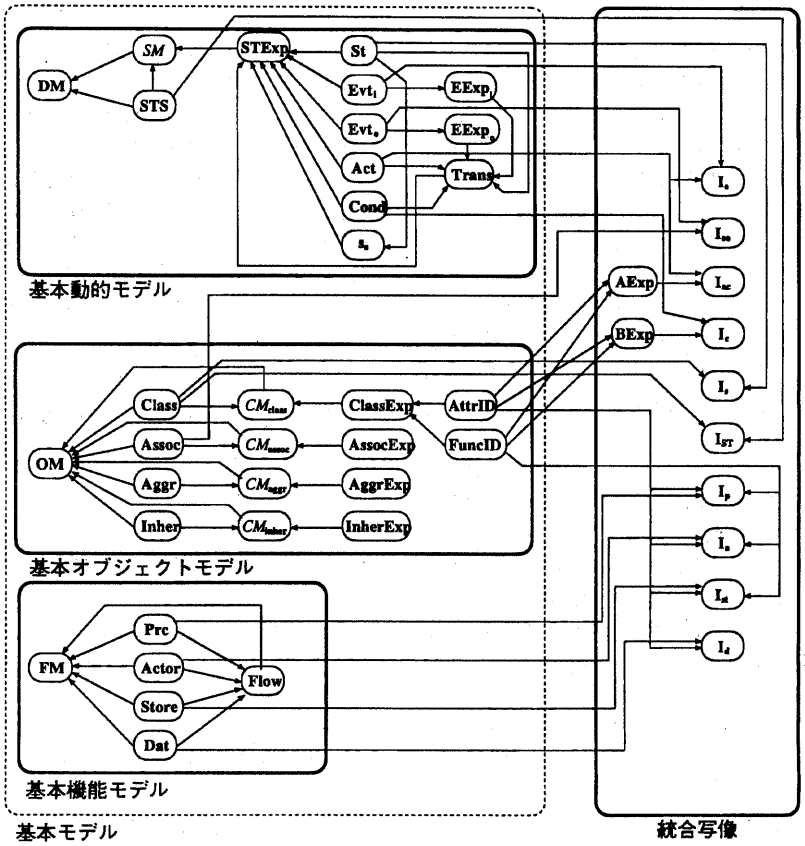

図2　各構成要素と統合写像の依存関係

$$DG_n \stackrel{def}{=} (\mathbf{C}_n, \mathbf{A}_n)$$
$$\text{where } \mathbf{A}_n = \{R_n(C_\alpha, C_\beta) | C_\alpha, C_\beta \in \mathbf{C}_n\}$$

Example

$cexp = (\{a, b\}, \{m\}) \in ClassExp_n$の時、

$$(a, cexp), (b, cexp) \in R_n(ClassExp_n, \mathbf{AttrID}_n)$$

■

定義 3.3　これらにより、構築されたn段階目のモデルの部分整合条件$SWFC_n$(n th Sub Well-Formed Condition) を以下のように定義する。

$$SWFC_n \Leftrightarrow \forall (M, N) \in \mathbf{A}.$$
$$R_n(M_n, N_n) \subseteq M_n \times N_n$$

■

$SWFC_n$は、n段階目に構築されたモデルにおいて要素の合成として定義された式などが出現する場合、用いた要素がその時点のモデルにおいて不足なく定義されていることを示している。従ってこの条件はモデルの段階的構築の際、ある時点の近傍でモデルが極大の情報量を持つ条件であり、中間段階で検証などを行う際の指標として用いることができる。

段階的構築を繰り返して行く過程で$SWFC_n$が$SWFC_{n-1}$から余計な要素を削除することにより達成されているかもしれない。この様な場合、モデルの構成要素の包含関係に関してバックトラックが生じ、前の段階のモデル構成要素と等しくなってしまい、モデル構成要素の単調増加性を保証できなくなってしまう。この様子を図3に示す。この図では時系列の$n-2$段階から$n-3$段階に移行する時、モデル構成要素の削除が行われ、その包含関係に関してバックトラックが生じている。そこで、ステップと言う尺度を導入する。ステップは、図のようなバックトラックが生じた場合、モデルの構成要素集合が等しい段階にまでさかのぼって数字を付け直すものである。このようにすると、v段階でのモデルは$v-1$段階のモデルに対して何か情報を付加したことになり、ステップ番号に対

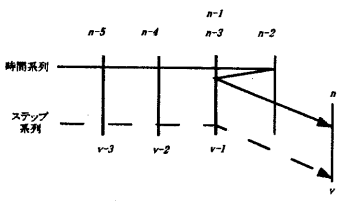

図3　時系列及びステップ系列で見たモデルの構築過程

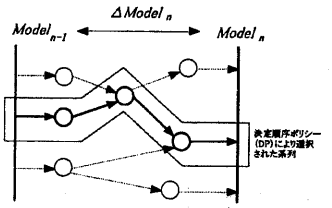

図4　ポリシーによる系列の選択

するモデル構成要素の単調増加性を保証することができる。

記法 3.2　以降、添字はステップを表現するものとする。

∎

このように、ステップが増大するにつれて、決定したモデル構成要素は増加し、最終的には目的とするモデルができあがる。この最終形の添字を ∞ とすると、最終的な整合条件を**完全整合条件** $CWFC$(Complete Well-Formed Condition) と呼び、以下のように定義する。

定義 3.4

$$CWFC \Leftrightarrow \forall (M,N) \in \mathbf{A}.$$
$$R_\infty(M_\infty, N_\infty) \subseteq M_\infty \times N_\infty$$

∎

3.2　決定順序関係

以上で定義した整合条件は、段階的モデル構築の際の静的な区切りを表現するものであった。又、$n-1$ ステップから n ステップを構築する活動は、モデル構成要素を増加させるものである。

記法 3.3　$n-1, n$ ステップのそれぞれのモデル全体を $Model_{n-1}$, $Model_n$ と書き、そのモデル構成要素の差分を $\Delta Model_n$ と書く。

∎

$\Delta Model_n$ は $Model_{n-1}$ から $Model_n$ を構築するにあたり、付け加えるモデル構成要素を表現している。このようなモデル構成要素を決定する順序はさまざまなものが考えられ、その列は前節で定義したステップ間を接続するものとなっている。この要素を決定する順序関係のことを**決定順序関係** ($\prec$) と呼び以下のように定義する。

定義 3.5　要素 a の後に要素 b を決定又は削除する時、$a \prec b$ とする。

∎

この決定順序関係は 1 直線の全順序を形成し、その列は $\Delta Model_n$ 順列分存在する。決定順序関係により、$n-1$ ステップから n ステップの**構築法** BM_n を以下のように定義する。

定義 3.6

$$BM_n \stackrel{def}{=} (\Delta Model_n, \prec)$$
where $\prec$: 全順序関係

又、すべての BM_n の集合を BMS_n と書くことにする。

∎

以上のように、FOVM における構築メカニズムは整合条件によるステップへの分割と、決定順序関係による構築法により特徴づけることができる。

3.3　ポリシーの導入

無造作にモデルを構築すると、余計な後戻りや回り道が生じたり、安定したモデルを構築できない場合が多い。そこで、モデルを決定する指針を導入する。このような指針は、モデル構築に関して制約を与えるものであり、ポリシーと呼ぶ。ポリシーには以下の 2 種類が存在する。

(1)　決定順序ポリシー
(2)　区切りポリシー

決定順序ポリシーは、ステップ間を接続する決定順序列を制限するもので、モデル構成要素の抽出順序を規定するものである。又、区切りポリシーは、モデル構築の目安であるステップを制限するものであり、モデルのチェックなどを行うチェックポイントを設定するものである。

3.3.1　決定順序ポリシー

構築法 BM_n は $n-1$ ステップから n ステップに至るまでの間に、どの様にモデル構成要素を決定するかを表現したものであった。このような構築法は、決定しなければならない要素 $\Delta Model_n$ の順列 ($|\Delta Model_n|!$ 通り) だけ存在する。このような構築法すべてを許可する場合、前述したような障害が生じる恐れがある。そこで、構築法を制限する決定順序ポリシー DP を導入する。

定義 3.7　決定順序ポリシー DP を以下のように定

義する。

$$DP : BMS \rightarrow Bool$$
$$\text{where } BMS = \bigcup_{1 \leq i \leq \infty} BMS_i$$

■

この決定順序ポリシーDPにより、ステップ$n-1$からステップnに至る構築法の集合BS_nは以下のように制約される。

$$BS_n = \{BM_n \in BMS_n | DP(BM_n)\}$$

例として、基本オブジェクトモデル中の継承関係式の構築に関して、「継承関係式はそれを構成する親クラス／子クラスが定義されなければ定義出来ない」と言う決定順序ポリシーを定義したとする。この決定順序ポリシーDPは以下のように表現することができる。

$$DP(BM_i) = \forall c_1, c_2, inherexp \in \Delta Model_i.$$
$$c_1 \in ClassID \wedge c_2 \wedge ClassID$$
$$\wedge inherexp \in InherExp \Rightarrow$$
$$c_1 \prec c_2 \prec inherexp \vee c_2 \prec c_1 \prec inherexp$$

この様な決定順序ポリシーは開発対象のシステムの性質によっても異なるものであり、開発毎にどのような指針で構築するかを決定して支援環境などで適切に選択出来るべきものである。

3.3.2 区切りポリシー

構築メカニズムで定義したステップは細かすぎて、実際のシステム開発ではほとんど意味の無い区切りしか表現しない。そこで、モデル構築の目標を決定するためにステップをある程度制限する必要がある。区切りポリシーはこのようなステップの制限を表現するものである。

定義 3.8 区切りポリシーSPを以下のように定義する。

$$SP : \{Model_i | i \in \mathbf{N}\} \rightarrow Bool$$
where $\mathbf{N}$: 自然数

ただし、以下の条件条件を満たさなければならない。

$$\forall i \in \mathbf{N}. SP(Model_i) \Rightarrow SWFC_i$$

■

このような区切りポリシーも決定順序ポリシーと同様、開発対象のシステムの性質により異なるものであり、支援環境で適切に取り扱う必要がある。

3.4 要素の削除

上記の方法に従って構築した要素を削除する場合には、適切な方針が必要となる。すでに$SWFC$を満たしているようなものからその構成要素の一部を削除した場合、それは構築途中段階に戻ったと考えられる。また構築途中のものであっても、その構成要素の削除が決定順序ポリシーに反していなければ、それが含まれる式などを完全に削除してしまう必要はない。

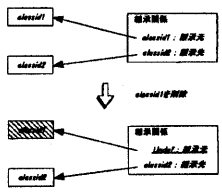

図5 継承関係における削除例

そこで、特別な識別子$Undef$を導入し、$Undef$が存在するモデル$Model_i$は$SWFC_i$が成立しないものとする。ある要素を削除した場合、そこは先ず一時的に$Undef$に置き換える。以下の継承関係を例にとる。

今、継承識別子$inherid \in$ **InherID** と継承式$inherexp$が以下の様に定義されているとする。

$$\mathcal{CM}_{inher}[ClassSet](inherid) = inherexp$$
$$inherexp = (c_1, (c_2))$$
$$where$$
$$ClassSet \subseteq \textbf{ClassID},$$
$$c_1, c_2 \in ClassSet,$$

この時クラス識別子c_1が削除された場合、この継承識別子が含まれるモデルの$SWFC$は壊れたことになる。しかし決定順序ポリシーで、どちらか一方のクラス識別子さえ定義されていれば継承関係式を構築できると定義されていた場合、途中過程に戻ったものとして、以下の様に定義する。

$$inherexp = (Undef, (c_2))$$

つまりクラス識別子を削除しても、継承関係式は残る。この様子を図5に示す。但し継承関係式に関する決定順序ポリシーが上記の様なものではなかった場合、クラス識別子の削除が継承関係式にも波及することは考えられる。

以上を基にして$Undef$を用いた削除の方針を決定する。

(1) 削除された部分には$Undef$を割り当てる。
(2) 最初に定義する時に未定義のままの残された部分についても$Undef$を割り当てる。
(3) ある要素を構成する部分が削除された場合、その状態が決定順序ポリシーに反しない限りその要素は削除されない。

4. 分析支援環境の実装

以上で定義された概念及びその要素間の関係を基に、

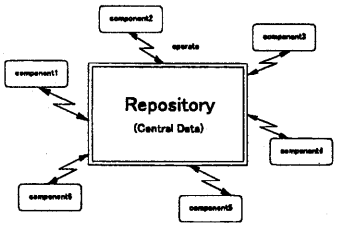

図6　リポジトリ・アーキテクチャスタイル

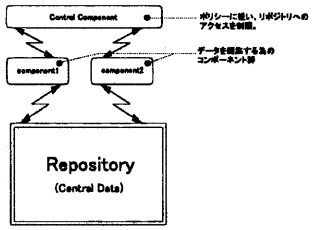

図7　制御コンポーネントの導入

FOVM を用いて分析モデルを構築するための支援環境を提案する。前節で定義した決定順序ポリシー及び区切りポリシーは支援環境においてどの部分から分析始め、どの様に分析を進めて何を目標にすれば良いかと言う指針を与える重要な部分である。

4.1　段階的構築を支援する環境

前記の通り、大規模なシステムの分析では一度に仕様中の全ての概念を抽出することは不可能である。従ってモデル構築は段階的に行われるが望ましい。途中段階では *SWFC* や区切りポリシーが満たされる訳ではなく、決定順序ポリシーに従ってその構築要素の中に *Undef* を含む様なものが存在する。このようなモデルは、分析が進むにつれて不完全な構成要素が減少するように支援を行わなければならない。よって支援環境はこの様な不完全な構成要素も含めて、何らかの形で情報を保持しておかなければならない。またその情報をもとに、分析者への支援を行い、その反応として情報を段階的に更新していく。

我々は、このような支援環境のアーキテクチャとしてリポジトリ・アーキテクチャスタイル[5]を選択した。このスタイルでは中心に位置する共有情報と、そのデータに対して操作を施す各々独立したコンポーネント群からなる。この支援環境の場合、中心に位置するものは上記の構築途中段階の要素も含めた情報であり、データに対して操作を施すコンポーネントは、各要素を編集するための手段である。このスタイルの一般的な形を図6に示す。

我々が構築する支援環境では、要素を編集する作業は決定順序ポリシーのため独立ではない。よって、リポジトリにアクセスするコンポーネントは独立ではなく、決定順序ポリシーによる制約を受けることになる。そこで、リポジトリにアクセスするコンポーネントを制御する制御コンポーネントを導入する。このような支援環境のアーキテクチャを図7に示す。

4.2　関係データベースの導入

リポジトリ・アーキテクチャのリポジトリに当たる部分には、関係データベースを当てはめる。これにより中間状態のモデル情報が保持され、またデータの永続性が確保出来る。

FOVM で定義されている各要素を関係データベースのスキーマにより表現する。例として継承関係式を表現するスキーマを考える。このスキーマで必要となる情報は継承関係式における親クラス／子クラスである。従ってそのスキーマは以下の表1の様な構成となる。

表1　データベース中の継承関係テーブル

src	dst
$classid_p$	$classid_c$

他の要素についても基本的にこれと同様に定義する。また、各スキーマには主キー (*Primary Key*) を設定し、データベース中での重複を防ぐ。例えば上記の継承関係式を表すスキーマでは主キーを *src*, *dst* に設定する。属性識別子集合のようにその中の識別子名に重複が許されるべきもの（複数のクラスで同じ属性の定義がなされているかもしれない）については、主キーとして識別子と意味をとることにする。また継承関係式が参照しているクラス識別子名はその識別子集合を表す表中で主キーとなるものなので、これを外部キー (*Foreign Key*) として定義する。

4.3　Java を用いた支援環境の構成例

以上で定義及び決定した方針に従って、支援環境の実際の構成例を示す。開発言語としては Java を使用する。Java を用いる利点としては以下が挙げられる。

- JDBC API によるデータベースとの結合性
- 分散開発の可能性
- マルチプラットホーム対応

支援環境の全体像を図??に示す。

先ず *Model Selector* を用いて編集するモデルを選択し、その情報を JDBC API に送る。データベースとの通信はこの API を通して行なわれる。実際に

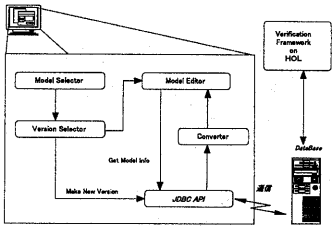

図8　支援環境の全体像

データベースに定義されているデータは文字列であるため、それを環境側で整形して必要な情報を表示する。

5. ま と め

本論文では、先ず形式的オブジェクト指向分析モデル FO∀M の概念を説明し、次に分析モデル構築の際に存在する関係及び規則を抽出した。これらは、例題に対してさまざまなアプローチで分析を行い、抽出したものである。又、それらがモデル構築メカニズムに対してどのような影響を与えるかを明らかにするために、モデル構築メカニズムと抽出した関係の形式化をおこなった。

部分整合条件 ($SWFC_n$) は段階的に構築されるモデル全体の、n ステップにおいて構築されている部分で満たされる静的な関係を示している。この様にして構築された最終的なモデルにおいても整合条件は守られているべきであり、これを完全整合性 $CWFC$ として定義した。

$n-1$ ステップ目のモデル $Model_{n-1}$ と n ステップ目のモデル $Model_n$ の間に決定しなければならない、$\Delta Model_n$ 個の要素の決定順序はその順列分存在している。これに対して決定順序ポリシー (DP) を導入した。これによりモデル構築時にその指針を与えることが出来、余計な後戻りなどを制限出来る。

要素を削除する際は $Undef$ を導入した。構築途中段階でのモデル中要素はこの様な識別子を用いて記述され得る。但しある要素を削除した場合にその前の段階は戻るかどうかは上記決定順序のポリシーによる。

以上の関係や指針に従って FO∀M の概念に沿った分析支援環境を提案した。この様に段階的に進められるモデル構築の中間情報を保持しておく環境のアーキテクチャとしてはリポジトリ・アーキテクチャスタイルが適切である。実際にそのリポジトリを実現する部分として関係データベースを導入し、その中での表定義について触れた。この様にしてデータベースに保存されるデータは分析以降のフェーズでも使用されるものである。

参 考 文 献

1) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-oriented modeling and design*, Prentice-Hall International (1991).
2) 青木利晃, 片山卓也: オブジェクト指向方法論のための形式的モデル, 情報処理学会 OO'96 シンポジウム (1996).
3) 青木利晃, 片山卓也: モデルの一貫性の検証のための公理系, ソフトウェア科学会第 3 回 ソフトウェア工学の基礎 ワークショップ (1996).
4) 青木利晃, 石田至, 古川順一, 片山卓也: オブジェクト指向分析モデルにおける一貫性検証のための公理系の実装, ソフトウェア科学会 第 14 回大会 (1997).
5) Shaw, M. and Garlan, D.: *Software Architecture*, Prentice-Hall (1996).