ソフトウェア工学 118－10
（１９９８．３．１０）

オブジェクト指向モデリング方法論「ＢＯＡＤ法」

ーオブジェクト・モデリングのフレームワークー

妻木俊彦、岩田裕道、森澤好臣

日本ユニシス株式会社

オブジェクト指向技術は、そのモデリング能力の高さと表現の分かり易さから、新しいシステム開発技術として期待されてきたが、アプリケーション・システムへの適用は予想したほど進んではいない。理由の１つとして、問題領域に関する知識とソフトウェア構築に関する技術の両立を図らなければならないオブジェクト・モデリングの難しさを挙げることができる。本報告では、人間の思考形式に基づいたモデル駆動アプローチ、ワークスペース・アプローチ、アーキテクチャ指向アプローチという３つのアプローチからなるオブジェクト・モデリングのフレームワークの提案と、それに基づいたオブジェクト・モデリング方法論「ＢＯＡＤ法」を紹介する。

A Framework and Methodology for Object Modeling

Toshihiko Tsumaki, Hiromichi Iwata, Yoshitomi Morisawa

Nihon Unisys, Ltd

Though object-oriented technology has been expected of new system development technology for the high ability of modeling and the understandability of models, the number of object-oriented applications does not increase. One of the reasons is difficulty of object modeling that make to join domain knowledge with software technology. This paper will introduce a framework for the object modeling and the object modeling methodology named BOAD based on the framework. The framework is consisted of three approaches named model driven approach, workspace based approach and architecture oriented approach.

１．はじめに

オブジェクト指向技術は、仕様の理解の容易さ、ソフトウェア部品の再利用の容易さ、作られたシステムの拡張性の高さなどによって注目されてきた。更に、仕様に関しては、最近、共通仕様記述言語ＵＭＬ[1]が発表され、その表現の一般性が高められつつある。しかし一方で、システム開発方法論、特にモデリング技術の一般化はなかなか進展しない。世に様々な方法論[2)3)4)]が問われているにも係わらず、モデラーと呼ばれる技術者の数は相変わらず少ないのが現状である。

オブジェクト・モデリングの「分かり易さ」は、そのモデル要素の抽象度の低さに負う処が大であり、そのことが反対にシステム仕様に曖昧さをを持ち込むことにもなっている。そして、研究の進展につれて、オブジェクト指向に関する技術は詳細化と厳密化の道をたどり始めた。このことはシステム化技術としての完成度を高めている反面、オブジェクト指向技術が本来持っていた、そしてそのことによって多くの人々の期待を集めてきた「分かり易さ」という性質を失わせることにもなりつつある。実世界の実体や現象の意味を取り扱うオブジェクト指向システム開発にとって、開発プロジェクトへの領域専門家の参画は必須である。こうした、非情報処理部門の業務専門家との共同作業を可能とするオブジェクト・モデリングの方法が問い直されなければならない。

オブジェクト指向技術が持っている「分かり易さ」という性質を保持するために、オブジェクト・モデリングという作業を人間の思考という側面から捉え直すことが有効ではないかという仮定に基づいて作成したオブジェクト・モデリングのフレームワークと、そのフレームワークに基づいた方法論「ＢＯＡＤ法」(Business Object Analysis and Design)の概要について報告する。

２．オブジェクト・モデリングとは

我々は、感覚器官を通して得られた様々の情報を統合し、再構築することによって実世界のイメージを作り上げ、それを記憶し、理解している。こうして得られた実体や現象に関する概念を認知モデルと呼ぶ。

オブジェクト・モデリングは、実世界に対して得ら

れたこの認知モデルを、その基本的な意味と構造を変えることなしに、オブジェクト指向パラダイムに基づいた、コンピュータ上で計算可能なモデルに変換する。すなわち、オブジェクト・モデリングとは実世界からコンピュータへとその作業空間を移動しながら順次モデルを変形してゆく作業である[5]。モデリングの基本的な概念を図1に示す。

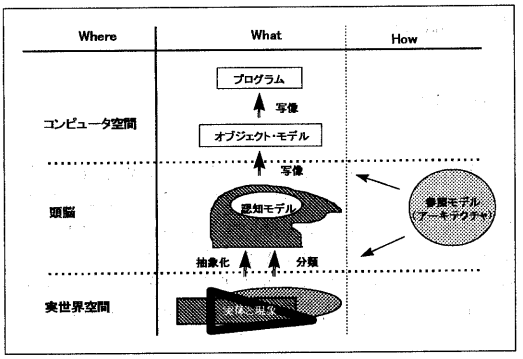

図1　モデリングの概念

図1で示したオブジェクト・モデリングの基本的な概念を2つの視点から考察してみることにする。

（1）文脈

人間は、言語によって理解する動物だと言われている。人間の理解を支えているのは、言語の中の文脈であり、この文脈は言語の違いを越えて、共通の枠組みを持っていると考えられている。

我々は、屡々、'what'、'who'、'when'、'where'、'how' などの疑問詞によって問題の内容を理解しようとする。これは、格文法[6]に於ける1種の格を示しているとも考えられる。

■ 'what' は、何をするのかを規定する対象格である。オブジェクト・モデリングは、実世界の中のオブジェクト・モデルの作成から始まり、コンピュータ上で計算可能なオブジェクト・モデルの作成でその作業を完結する。即ち、オブジェクト・モデリングとは、モデルを作成する作業である。

■ 'where' は、それを何処で行うのかを規定する場所格である。オブジェクト・モデリングは、実世界とコンピュータという2つの空間でそれぞれのモデルを順次、作成していると言うことができる。

■ 'who' は、誰がそれを行うかを規定する動作主格である。オブジェクト・モデリングの作業者は、それぞれの作業空間でのモデリングの専門家である。つまり、作業者は必用なモデルを作るための作業空間が指定されることによって自動的に決定される。

■ 'when' は、それが何時行われるかを規定する時間格である。ラウンド・トリップ型の開発プロセスであるオブジェクト・モデリングの作業順序は時間によって規定されるのではなく作業内容に依存する。すなわち、作業順序はその作業を行う作業空間間の移動として規定される。

■ 'how' はどのようにそれを行うかを表している道具格である。モデリングの道具は人間の思考能力であるが、その思考の中核には過去の知識に基づいた何らかの参照モデルがあるものと考えられる。このモデルをシステム的に表現したものをアーキテクチャと考える。

このように、文脈という視点から捉えたオブジェクト・モデリングを規定する基本的な枠組みは、対象と場所と道具の各格によって表されるところのものである。

（2）能力

方法論があっても、モデリングは、人間が持っている基本的な思考能力に依存するところが大である。こうした人間の思考を支えている様々な能力の中から、抽象化と分類と写像の3つの能力をモデリングにとっての基本能力と考える。

■抽象化能力は、我々が時空間軸の中で認識した対象に関する情報の中から必用な情報だけを選択し、その本質を理解し、モデルを作成することができる能力である。

■分類能力は、選択的に認識された情報同士の間の意味的な同質性や異質性を判別し、統一的なモデルを構成する能力である。

■写像能力は、実世界のモデルをコンピュータ上で計算可能なモデルに変換する能力である。モデルの変換は、異なった空間の間でのモデル構成要素の写像として捉えることができる[7]。

3．オブジェクト・モデリングのフレームワーク

前記2つの視点に基づいて、オブジェクト・モデリングの3つのアプローチを以下のように設定した。

(1)モデル駆動アプローチ

オブジェクト・モデリングとは、実世界に対する認知モデルを順次変換しながら、最終的にはプログラム・コードを作成する作業である。この変換過程で、様々なモデルが作成される。モデルを作成するのは抽象化能力である。モデル駆動アプローチは、システム開発をモデルを作成するプロセスと捉え、そこで作成するモデルを定義することによってそのプロセスを定義しようとするアプローチである。

(2)ワークスペース・アプローチ

システム開発とは、実世界モデルのコンピュータ空間への写像である。認知モデルは、幾つかの作業空間での写像を繰り返すことによって、コンピュータ空間への写像を完成する。各作業空間では、モデル変換によって新たなモデルが作成されてゆくことになる。ワークスペース・アプローチは、このような作業空間を規定することによってモデル変換作業を定義しようとするアプローチである。各作業空間には、そこで作成するモデルと、そのモデルの作成者およびモデル変換作業手順などが定義される。

(3)アーキテクチャ指向アプローチ

優れたモデルは、優れたアーキテクチャを持っている。言い換えれば、優れたアーキテクチャに基づいて作成されたモデルは、優れたモデルであると言うことができる。アーキテクチャは、システムを構成している各要素の機能構造と要素間の制御構造によって規定される。オブジェクト指向アプリケーション・システムのアーキテクチャは、それを構成している各クラスの機能分類が基になっている。アーキテクチャ指向アプローチは、こうしたアプリケーション・アーキテクチャに基づいてオブジェクト・モデルを構築しようとするものである。

4．モデル

モデリングとは、実世界をモデル化することではない。それは、人によって認識された実世界のモデルを仕様化することである。これを認知モデルと呼ぶ。すなわち、モデリング作業とは、人間の認知モデルを、抽象化、分類、写像といった人間の思考能力による変換操作によって最終成果物であるプログラム・コードに変換するプロセスである。　オブジェクト・モデルの体系図を図２に示す。網部をモデリングと呼ぶ。

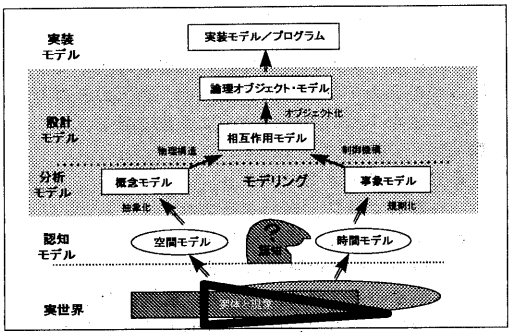

図２　オブジェクト・モデリングの体系

それぞれの操作によって作成されるモデルは、以下の通りである。

■認知モデル

同一の実体であっても、それを見る人間の視点によってものの見え方は異なる。しかし、一方で人間には人間独自の認識の枠組みがあると考えられている。時間と空間は、我々人間の認知形式に基本的な枠組みである。即ち、感覚器を通して実世界から得られた各種情報は、空間軸と時間軸の上で再構築されイメージとして記憶される。即ち、我々の認知モデルは、空間モデルと時間モデルという１組のモデルから構成されていると考える。認知モデルは、我々の脳の記憶機構の中にあるモデルである。

■分析モデル

分析モデルは、認知モデルである空間モデルと時間モデルを定式化したモデルであり、概念モデルと事象モデルという１組のモデルから構成されている。

概念モデルは、空間モデルをビジネスという視点から抽象化したモデルである。問題領域の構成要素とその属性及び構成要素間の関係を、オブジェクト指向パラダイムに従って仕様化する。概念モデルは、実世界の構造を表現したモデルである。

事象モデルは、時間モデルをビジネスの視点から規則化したモデルである。時間の経過に従って生起し、変化、消滅する事象と、事象間の関係を仕様化する。事象モデルは、実世界の因果関係を表現したモデルである。

■設計モデル

設計モデルは、コンピュータ上で計算可能なモデルである。設計モデルには、相互作用モデルと論理オブジェクト・モデルがある。

相互作用モデルは、実世界を時間軸と空間軸によって分割した概念モデルと事象モデルをコンピュータ・システム上で再統合したモデルである。すなわち、概念モデルで描かれた物理構造に事象モデルで描かれ因果律を制御構造に変換して統合することすることによって、コンピュータ上で計算可能なモデルとする。

論理オブジェクト・モデルは、相互作用モデルによって描かれた計算可能なモデルをオブジェクト指向パラダイムによって表現した論理的なオブジェクト・モデルであり、問題領域のコンピュータ・システム上への写像である。

■実装モデル

実装モデルは、論理オブジェクト・モデルに実装言語の制約を作用させたモデルである。実装モデルは、実装設計モデルとプログラム・コードである。プロトタイピングによって、論理オブジェクト・モデルから直接プログラム・コードを作成する場合は、実装設計モデルが描かれない場合がある。

5．ワークスペース

モデリング作業は、そのモデルが属する作業空間で行われる。そして、モデルの変換とは、異なった作業空間の間でのモデルの写像である。モデルとワークスペースの関係を図３に示す。網部がモデリングである。

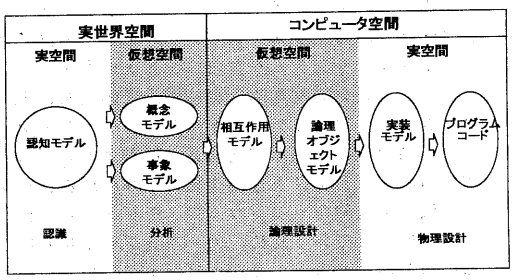

図３　モデルとワーク・スペース

モデリングのための作業空間として、まず、実世界空間とコンピュータ空間という２つの空間が考えられ、

更に、これら２つの作業空間をそれぞれ実空間と仮想空間に分ける。

■実世界の実空間

問題領域の専門家やシステムの利用者実世界の実空間、即ち、実際のビジネス領域の中に存在する実体やそこで起きている現象を認識し、その意味を定義する。この作業空間は、業務専門家やシステムの利用者の頭の中にある空間で、そのモデルは認知モデルである。

■実世界の仮想空間

問題領域専門家が抱いている認知モデルは、分析者によってモデルとして定式化され記述される。この作業空間は、分析者の頭の中に描かれた実世界に対する仮想空間であり、そこで作成されるモデルは概念モデルと事象モデルである。この２つのモデルを会わせて分析モデルと呼ぶ。

■コンピュータ上の仮想空間

実世界に関する分析モデルは、設計者によって、コンピュータ上で計算可能な設計モデルに変換される。ここで設計者が前提とするのは、このモデルを実際に実装するコンピュータシステムではなく、仮想的なコンピュータ・システムである。そこで作成されるモデルは、相互作用モデルと論理オブジェクト・モデルである。

■コンピュータ上の実空間

論理オブジェクト・モデルは、実際のコンピュータ上で実行可能な実装モデルへの変換を通してプログラム化される。そのモデルの実装環境用の実装モデルへの変換は、実装環境を熟知している実装者の役割である。

６．アーキテクチャ

優れたシステムを作成するためには、優れた手本を基に独自の工夫を凝らすことが重要である。コンピュータ・システムの手本とは、ソフトウェアの構造と振る舞いの基本的な枠組みを示してくれるものである。即ち、優れたシステムとは優れたアーキテクチャに基づいたシステムである。

コンピュータ・システムのアーキテクチャには、システム・アーキテクチャとアプリケーション・アーキテクチャがある。システム・アーキテクチャはアプリケーション・システムを含むコンピュータ・システム全体の構成要素の物理的な機能構造を規定し、アプリケーション・アーキテクチャはアプリケーション・システム内部の論理的な機能構造と制御メカニズムを規定する。オブジェクト指向技術により作成されたシステムは、自立的なソフトウェア・モジュールの集合として作成されているため、アーキテクチャ化し易い構造をしている。このようなオブジェクト指向技術の特徴を生かすためにも、優れたアーキテクチャの設計がシステム開発プロセスの中で重要な位置を占める。

６．１　クラス・カテゴリー

オブジェクト指向アプリケーション・システムには、様々なオブジェクトが含まれている。これらの各オブジェクトを、それぞれ類似した性質を持ったオブジェクト群に分類する。これをクラス・カテゴリーとする。このクラス・カテゴリーは、各クラスのシステム上の役割を基にしており、ビジネス規則にもとづいた機能分類ではない。図４にクラスの分類と、それに基づいたクラス・カテゴリーを示す。

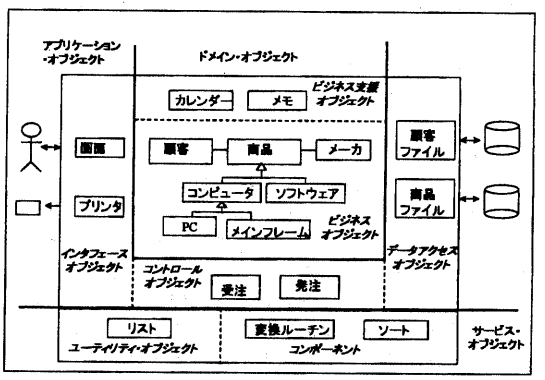

図４　クラス・カテゴリー

アプリケーション・システムを構成しているクラスは３つの大カテゴリーに分類でき、各大カテゴリーは、更に小さな７つのカテゴリーに分類できる。

■ドメイン・オブジェクト

◆ビジネス・オブジェクト

－顧客クラス、商品クラスなど

◆ビジネス支援オブジェクト

－情報の記録保管媒体（記録やメモなど）

－事象の発生源（時計、暦、監視員など）

－情報同士の関連構造（関連オブジェクト）

■アプリケーション・オブジェクト

◆インタフェース・オブジェクト

－画面クラス、印書クラスなど

◆コントロール・オブジェクト

－受注クラス、預金クラス、予約クラスなど

◆データアクセス・オブジェクト

－顧客ファイル、商品ファイルなど

■サービス・オブジェクト

◆ユーティティ・オブジェクト

－フィルター、コンテナ・オブジェクトなど

◆コンポーネント

－サブシステムや各種コンポーネント

（１）ドメイン・オブジェクト

ドメイン・オブジェクトは、現実世界の問題領域を構成している実体や現象を写像しているクラス群である。ドメイン・オブジェクトは、その性質により、ビジネス・オブジェクトとビジネス支援オブジェクトの２つのタイプに分類される。

■ビジネス・オブジェクト

ビジネス・オブジェクトは、問題領域の中に、対応する具体的な実体が実際に存在し、且つ、ビジネス活動に主体的に関与しているクラス群である。ビジネス・オブジェクトは、通常、管理すべき情報、すなわち、アクタが利用する多量の管理情報をインスタンスの属性値として保有している。したがって、多くの場合、ビジネス・オブジェクトは永続オブジェクトであり、これらの属性情報は、実装時にデータアクセス・オブジェクトを介してデータベースに保管されることになる。

■ビジネス支援オブジェクト

ビジネス支援オブジェクトは、問題領域の構成要素ではあるが、ビジネス活動に能動的に関与することはなく、むしろビジネス・オブジェクトを支援することによって間接的にビジネス活動に参画しているクラス群である。保有している属性の種類や量によっては永続オブジェクトになる場合もある。

（2）アプリケーション・オブジェクト

アプリケーション・オブジェクトは、ドメイン・オブジェクトをコンピュータ上で実行するために必要なクラス群である。アプリケーション・オブジェクトは、インタフェース・オブジェクト、コントロール・オブジェクトおよびデータアクセス・オブジェクトからなる。

■インタフェース・オブジェクト

インタフェース・オブジェクトは、アプリケーション・システムがシステムの利用者であるアクタとの間で会話を行うための情報通路であると共に、アクタや入出力機器などのフロントエンドをオブジェクト指向でラッパリングするためのクラス群である。

■コントロール・オブジェクト

コントロール・オブジェクトは、各ビジネス・オブジェクトのサービス機能を使ってビジネス機能を実現するクラス群である。コントロール・オブジェクトは、また、ビジネス規則をカプセル化することによって、ビジネス規則の変更に対し変更箇所を局所化し、システムの拡張性を高める役割を持つ。

■データアクセス・オブジェクト

データアクセス・オブジェクトは、永続オブジェクトとデータベース間のデータの入出力を支援する。すなわち、データアクセス・オブジェクトは、アプリケーション・システムを構成している永続オブジェクトの各インスタンスが持っている属性の値を外部記憶装置に保管するのを支援するためのクラス群であると共に、データ管理システムやデータベースなどのシステムのバック・エンドをオブジェクト指向パラダイムでラッパリングする。

（3）サービス・オブジェクト

ドメイン・オブジェクトやアプリケーション・オブジェクトを支援するモジュール群をサービス・オブジェクトとする。サービス・オブジェクトには、ユーティリティ・オブジェクトとコンポーネントがある。ユーティリティ・オブジェクトは各種ライブラリーやデータ構造のようにデータ処理機能を支援するための補助的なクラス群であり、コンポーネントは実行可能な共有ソフトウェア群である。サービス・オブジェクトは、再利用部品と言うことができる。

6．2　制御メカニズム

制御メカニズムは、各カテゴリー間での相互作用の方向を規定する。図5に制御メカニズムを示す。

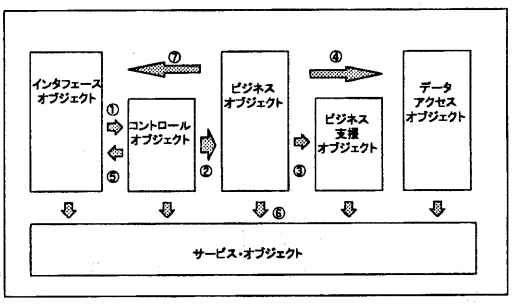

図5　制御メカニズム

図5の制御メカニズムを情報の流れに沿って説明する。説明の○数字は、図中の番号に対応する。

①インタフェース・オブジェクトによって受け取られたアクタからの入力情報は、インタフェース・オブジェクトによってシステムの内部コードや内部コマンドに変換された後、コントロール・オブジェクトに渡される。

②コントロール・オブジェクトは、ビジネス・ルールに基づいてビジネス・オブジェクトを順に起動しながら要求された仕事を遂行する。

③ビジネス・オブジェクトは、必要なら、ビジネス支援オブジェクトを使って、情報の一時保管、特定情報の参照などのビジネス・ルールの補完的な処理を依頼する。

④データアクセス・オブジェクトは、ビジネス・オブジェクトからの要求によって起動され、必要な情報をデータベースとの間で授受する。

⑤ビジネス・オブジェクトによる入力情報の処理の結果は、コントロール・オブジェクトに戻され、アクタへ応答するために該当するインタフェース・オブジェクトに渡される。

⑥サービス・オブジェクトは、他のオブジェクトから直接呼び出され、システム内での共有機能を提供する。

⑦ビジネス・オブジェクトの内部状態をディスプレイ装置に表示する場合には、ビジネス・オブジェクトが該当するインタフェース・オブジェクトを直接起動する。

6．3　アーキテクチャ

クラス・カテゴリーと制御メカニズムを統合してアプリケーション・アーキテクチャとする。アプリケーション・アーキテクチャを図6に示す。

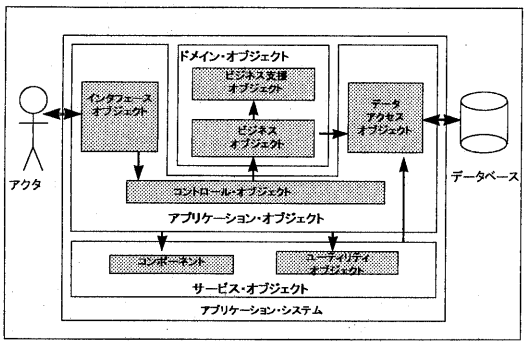

図6　アプリケーション・アーキテクチャ

7．「BOAD法」と事例

方法論は、モデル化のための技法と手順と表記法を規定したものである。

我々は、これまで述べてきたオブジェクト・モデリングのフレームワークに基づいた方法論「BOAD法」を開発し、その有効性を検証してきた。

「BOAD法」の基本的なコンセプトは、以下の通りである。

■思考支援

オブジェクト・モデリングは人間の思考そのものであり、方法論はその思考を支援するものでなければならない。

■領域特化

対象領域をビジネス・アプリケーションに絞ることによって、使い易い方法論とする。

■汎用モデル

論理設計と実装設計を分離し、実装ツールに影響されない汎用的なモデルを設計する。

7．1　システム開発の枠組み

システム開発を、図7で示す3つの工程からなるとする。ここで、「BOAD法」が対象とするのはモデリングのプロセスである。

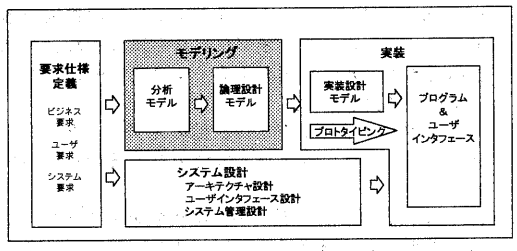

図7　システム開発の枠組み

（1）要求仕様定義

作成しようとするアプリケーション・システムに対する要求仕様を定義するプロセスである。アプリケーション・システムへの要求は、以下のようなものからなる。

■ビジネス・ニーズ

ビジネス規則に基づく要求

■システム・ニーズ

コンピュータ・システムに対する要求

■ユーザ・ニーズ

システムの利用者からの要求

（2）モデリング

実世界のビジネスからコンピュータ上で計算可能なモデルを作成するプロセスである。ここでは、次の2種類のモデルを作成する。

■分析モデル

実世界のモデル

■論理設計モデル

コンピュータ上で計算可能なモデル

（3）実装

実装言語の仕様にしたがった実装設計モデルの作成やプログラミング作業は、各実装環境に依存したプロセスとなる。

■実装設計モデル

実装環境上で実行可能なモデル

■プログラムとユーザ・インタフェース

実装モデルの実現

（4）システム設計

実際のシステム開発では、モデリング・プロセスに並行してシステム設計が行われる。システム設計には、以下のようなものの設計が含まれる。システム設計の多くは、オブジェクト指向に独自なものではない。

■アーキテクチャ設計

オブジェクトやデータベースの分散トポロジー、通信ネットワーク構造など

■ユーザ・インタフェース設計

インタラクション・インタフェース、ビュー・インタフェースなど

■システム管理機能設計

例外処理、運用管理など

7．2　「BOAD法」の概要

「BOAD」法は、分析モデルと論理設計モデルの作成を支援する。各モデルを作成するプロセスを、それぞれオブジェクト分析、論理オブジェクト設計と呼ぶ。各プロセスは、更に小さなステップから構成されている。即ち、それぞれのプロセスは、図8に示すように、該当するモデルを作成すの様々な技法を抱えており、この技法の適用のための手順を示してある。

（1）オブジェクト分析

オブジェクト指向による問題領域の分析をオブジェクト分析と呼ぶ。オブジェクト分析は、概念モデルと事象モデルによって問題領域をモデル化し、その構造を解析する。オブジェクト分析への入力は、認知モデルと要求仕様である。

■概念モデル作成

問題領域専門家が抱いている空間モデルから概念モデルを作成する。概念モデルを作成するために、

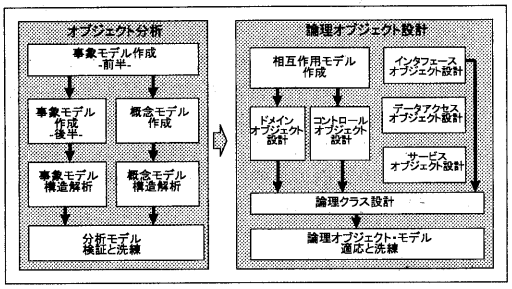

図８　「ＢＯＡＤ法」のプロセス

ドメイン分析法を用いる。問題領域を構成している実体の役割と責任をもとに、ドメイン・オブジェクトの各クラスを識別し、クラスの属性及びクラス間の関係を定義する。作成した概念モデルは、クラス図によって表記する。

■事象モデル作成

同様に、問題領域専門家の時間モデルから事象モデルを作成する。事象モデルを作成するために、ユースケース分析法を用いる。ユースケース分析は、システムが実現すべき機能を定義し、それを時間軸にしたがってシナリオとして記述する。作成された事象モデルは、ユースケース図とユースケース・シナリオによって表記する。

■構造解析

概念モデルと事象モデルの構造を解析し、共通要素や可変要素を抽出する。共通要素とは複数のクラスあるいはユースケースによって共有可能な要素であり、共通要素をカプセル化することによってソフトウェアの部品化、再利用を促進する。可変要素とは、ビジネス規則や実装ツールの変化によって大きな影響を受ける要素であり、可変要素をカプセル化することによって外的要因によるソフトウェアの変更を局所化する。

■検証と洗練

分析モデルは、要求仕様を満足しているかどうかを検証する。また、システムの拡張性を十分に考慮してモデルの基本構造を洗練する。

（２）論理設計

論理設計モデルは、実装ツールから独立した設計モデルである。論理設計では、アプリケーション・システムとしての構造を設計し、各クラスの操作や属性などを定義することよってシステムを構成しているオブジェクトを完成する。

■構造設計

構造設計は、概念モデルと事象モデルの統合と必用なクラスの追加によってシステムとしての基本構造を決定する。即ち、ドメイン・オブジェクトに加え、コントロール・オブジェクトやインタフェース・オブジェクト、データアクセス・オブジェクトを新たに識別し、事象モデルを使って相互作用モデルを作成する。相互作用モデルは、コンピュータ上に再現されたビジネス世界ということができる。相互作用モデルは、シーケンス図によって表記する。

■論理クラス設計

論理クラス設計では、相互作用モデルをもとに各クラスにサービスを割り当て、このサービスから操作を定義する。また、属性のデータ構造を定義する。論理オブジェクト・モデルは、クラス図によって表記する。

■適用と洗練

作成された論理オブジェクト・モデルの効率を予測し、必用ならモデルの修正を行う。また、可能なら設計パターンの適用を試みる。

７．３　分析設計手順

既に述べたように、オブジェクト思考モデリング作業は作業空間の中を渡り歩くものであって、時間軸に従って進められる訳ではない。これは、ラウンドトリップ型の手順と呼ばれている。図９にラウンドトリップ型手順の概念図を示す。

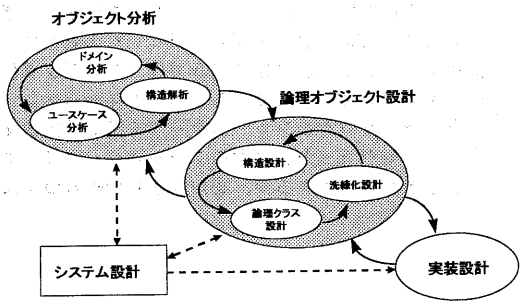

図９「ＢＯＡＤ法」のラウンドトリップ手順

７．４　事例

本方法論は、これまで以下のようなシステムの開発に適用してきた。

・予約型システム

予約型システムは、受発注型システムと共に主要なビジネス・アプリケーションの１つである。その基本形式は、領域と対象と時間の３項関係によって表現される。

・配信型システム

配信型システムは、顧客サービス・アプリケーションであり、交錯する情報構造とシステムの効率と拡張性の調和がモデリングの焦点となる。

・ワークフロー型システム

ワークフロー型アプリケーションは対象管理のアプリケーションであり、情報管理型のアプリケーションとは、モデリングの視点が異なる。

８．考察

実際のアプリケーション・システムへの適用を通して、今回のオブジェクト指向モデリングの枠組みの有効性は、ある程度確認できた。しかし、今回の方法論

開発とその適用にあたって、以下のような問題点を認識した。

・汎用的な方法論は、モデリング作業を難しくするだけである。オブジェクト指向パラダイムはビジネス・アプリケーションにとって非常に有効であるが、エンジニアリング・アプリケーションには必ずしも十分とは言い難い。領域を限定することによって、分かり易い方法論の実現が可能である。

・統一モデリング言語‘ＵＭＬ’はオブジェクト指向技術者が共有できるモデリング言語であるが、そのカバーする範囲は広く、また、必ずしも十分なモデル要素が揃っているわけでもない。必用なモデル要素のＵＭＬからの切り出しと拡張が必要である。

・シームレスという名の下に安易なモデリングが行われ、幾つかの失敗例が取りざたされている。外部仕様と内部仕様という視点で方法論全体を見直してみる必要がある。

９．おわりに

オブジェクト・モデリングは、人間の思考を支援するものであるにも係わらず、モデリングの難しさが依然解消していない。今回の試みでは、モデリング作業そのものは人間の思考であるという立場からオブジェクト・モデリングのフレームワークの設定と方法論の作成・実施を行った。今回のフレームワークは、まだまだ稚拙なものであるが、認知科学的枠組みの導入がオブジェクト・モデリングの難しさを解決する１つの方向性を示しているという感触を得ている。オブジェクトという豊かな、それ故曖昧な素材を有効に使って、厳密なコンピュータ上のアプリケーション・システムを作成するためのモデリングのフレームワークと方法論に対する１つの試みについて報告した。今後の課題として、より人間の思考に近いフレームワークの作成を目指して行きたい。

参考文献

１）ＵＭＬ： http://www.rational.com/uml/index.shtml

２）Rumbaugh,j., Blaha,M., Premerlani,W., Eddy,F., and Lorensen,W.: *Object-oriented modeling and design*, Prentice Hall, 1991

３）Jacobson, I: Object-Oriented Software Engineering, Addison Wesley, 1992

４）Booch,T.: *Object-Oriented Analysis and Design with Application*, 2nd edition, The Benjamin / Cummings Publishing, 1994.

５）妻木俊彦他：*オブジェクト・モデル導出に関する一考察*, 情報処理学会研究報告, 96-SE-112

６）Filmore,C.: *Towerd a Modern Theory of Case & other ARTICLES*, Prentice Hall etc, 1975.

７）Fauconnier,G.: *Domain and Connections*, 認知科学の発展, Vol.3, pp.1-28, 1990.