ビジネス・アプリケーション向け オブジェクト指向モデリング技法「BOAD 法」

妻 木 俊 彦

1. はじめに

オブジェクト指向システム開発方法論といえば，いわゆる 3 大方法論をはじめとして数え切れないほどの方法論が提案されている．それらの方法論の多くはオブジェクト指向パラダイムに則ったシステムの構築を目標としているが，筆者の知る限り，その方法論によって構築されるシステムがどのような特徴をもつのかについて述べているものは少ない．しかし，オブジェクト指向パラダイムに則れば，それだけでオブジェクト指向が持っている様々な便益をそのまま享受できるとはとても思えない．明確な目標や意図を持たずに作成したシステムが予期せぬ効果を現すなどと言うことは希有なことであろう．

BOAD 法は，Business Object Analysis and Design の略であり，名前が示すとおりビジネス・アプリケーションを対象としたオブジェクト指向モデリング技法であり，日本ユニシスが独自に開発した方法論である．

2. BOAD 法の設計目標

BOAD 法の開発に当たって，三つの目標を設定した．

・拡張性の高いアプリケーション・システムの構築

・使いやすい方法論

・多くの人に理解され，長期間その価値を失わない成果物の作成

それぞれの目標が，BOAD 法の中でどのようなかたちで実現されているかについて，以下に紹介する．

2.1 システムの拡張性

オブジェクト指向アプリケーション・システムの特徴は，保守の生産性にあると言われている．ここで言う保守性とは，外的な条件の変更に如何に容易に対応できるかということを指しており，この保守の生産性を支えるのがソフトウェアの品質要因の中の拡張性である．すなわち，拡張性とはソフトウェア製品が仕様の変更に容易に適応できる能力[1]のことである．拡張性には二つの側面がある．その一つはシステムの変更の容易さであり，もう一つはシステムの基本構造の頑健さである．

システムの変更の容易さを高めるためには，システムを取り替え可能なモジュールによるビルディング・ブロック構造にしたり，変更可能性の高いホットスポットを他の部分から分離独立させることが有効であると言われている．特に，問題領域の中でより多く変更されるのは，情報構造よりも，いわゆるビジネス活動である．オブジェクト指向システムでは，一連のビジネス活動の系列は，異なったオブジェクトの中を貫いて実現されることになる．それゆえ，仕様変更への対応の容易性を実現するには，

情報構造をもとにしたオブジェクトによる構造化に加えて，ビジネス活動の系列に基づいた構造化[2]が図られる必要がある．

仕様の変更に強いシステムとは，その変更によってシステムの基本構造までが影響を受けることの少ないシステムのことである．アプリケーション・システムの基本構造を決定するのはアプリケーション・アーキテクチャであり，頑健なシステムを構築するには優れたアーキテクチャにもとづいたシステム構築が不可欠である．BOAD法の参照モデル[3]は，アプリケーション・アーキテクチャのためのフレームワークであり，アプリケーション・システムを構築する各クラスのカテゴリと制御メカニズムを規定している．個々のシステム開発プロジェクトは，この参照モデルをもとに独自のアプリケーション・アーキテクチャを設計することになる．

2.2 使い易い方法論

その技法がいかにマニュアル的，機械的なものであっても，それが使う人に不自然な思考を強要するものであれば，その技法はすぐに陳腐化してしまうに違いない．むしろ，創造的かつ発見的な作業であるモデリング作業を支援するモデリング技法は，人の思考を支援するものでなければならない．より多くの人が自然で無理なく使用できるモデリング技法とは，人の認知プロセスに沿った手法とプロセスを持った技法である．BOAD法では，人の基本的な認知基盤である空間と時間という枠組みの中で対象を捉え，その概念構造をオブジェクト・モデルにマッピングしている．

アプリケーション・システムは，その適用領域によってビジネス・アプリケーションとエンジニアリング・アプリケーションに分類できる．ビジネス・アプリケーションとは，実世界を構成している実体や事象の振る舞いを情報によって表現し，管理する情報管理型のアプリケーションであり，エンジニアリング・アプリケーションとは，実世界のモデルをコンピュータ上で仮想的に実行するシミュレーション型のアプリケーションのことである．ビジネス・アプリケーションとエンジニアリング・アプリケーションではシステムの使用法が違うだけでなく，その開発方法にも大きな相違がある．単一の方法論によってこの両方のアプリケーションの構築方法を包含しようとすれば，その方法論はそれだけ多くのモデリング技報や表記法を必要とすることになる．そのことは，方法論を使う側にとっては負担にこそなれ決して益になるとは思われない．BOAD法は，その適用範囲をビジネス・アプリケーションに限定することによって，簡潔で使い易いモデリング技法を実現した．

2.3 モデリング成果物の恒久性

急激な進歩を続ける情報技術の発展のなかで，容易に陳腐化することのないモデリングの成果物が求められている．すなわち，実装環境に依存したモデルは，低価格で高性能の新しい情報機器の出現と共にその価値を失うことになり，独善的な表記法で記述されたモデルは，他の人にとって理解不可能なモデルとなる．

論理設計と実装設計を分離することによって実装環境から独立したモデルを作成することが可能となる．論理設計で作成された論理モデルは，実装環境が決定された後，その環境に適応させるために独自の実装設計によって実装モデルに変換される．このように，論理設計と実装設計を分離することによって，ソフトウェアの実装環境を選定する自由度も高くなる．

モデリング成果物の永続性を高めるもう一つの方法は，モデルの記述に際して標準的な表記法を採用することである．幸いに，つい最近，OMG（オブジェクト・マネージメント・グループ：オブジェクト指向技術に関する国際標準化団体）によって標準表記法である UML（Unified Modeling language）[4]が制定された．UML を採用することによって，世界中のシステム技術者が理解可能なモデルを記述することが可能になり，また，UML で記述したモデルは市販のケースツールを自由に選択することができるようになる．

3. 参照モデル

堅牢なシステムとは，要求仕様や実装環境などの外部環境の変化に対しシステムの変更が局所化され，システムの基本的な構造への影響を極小化できるようなシステムである．このようなシステムを構築するためには，優れたアーキテクチャにもとづいたシステムの構築が欠かせない．ここでいうアーキテクチャとは，システムの基本的な構造を定義する規範のことである．アーキテクチャには，コンピュータ・システムを構成しているハードウェアやソフトウェアの論理的な構造や物理的な配置を定義したシステム・アーキテクチャと，アプリケーション・システムの論理的な機能構造と制御メカニズムを定義したアプリケーション・アーキテクチャがある．

BOAD 法が提供する参照モデルは，ビジネス・アプリケーション向けのアプリケーション・アーキテクチャのフレームワークである．参照モデルはシステムを構成しているオブジェクトをその論理的な機能によって分類し，それらの間の制御メカニズムを規定している．このオブジェクトの分類をオブジェクト・カテゴリーと呼んでいる．参照モデルは，あくまでアプリケーション・アーキテクチャのフレームワークであり，具体的なアプリケーション・システムの開発に際しては，この参照モデルをもとに，各プロジェクト毎に独自のアプリケーション・フレームワークを設計する必要がある（図 1）．

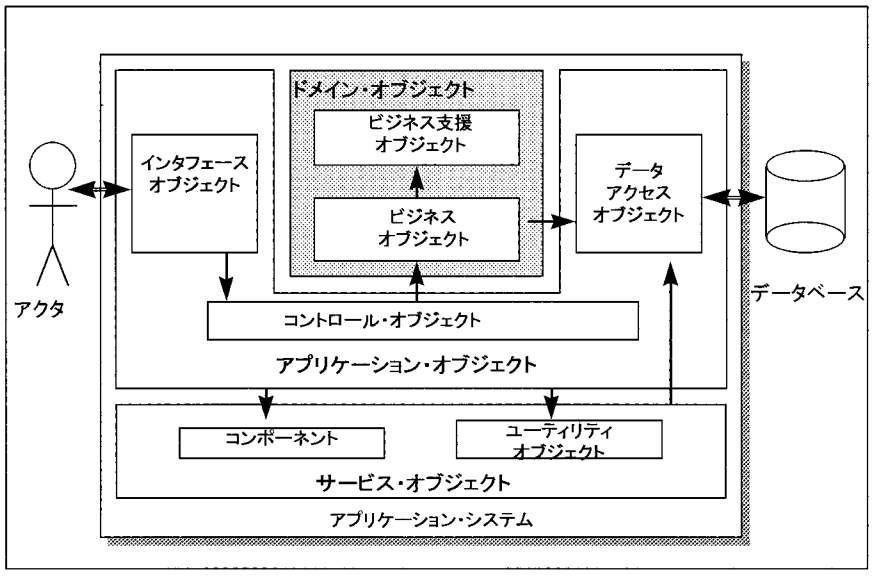

図 1　アプリケーション・フレームワーク

参照モデルのオブジェクト・カテゴリーは，以下の通りである．

ドメイン・オブジェクト

ドメイン・オブジェクトは，実世界の中で問題領域を構成しているオブジェクトである．ドメイン・オブジェクトは，ビジネス活動に直接関与している**ビジネス・オブジェクト**と，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**ビジネス支援オブジェクト**に分類される．ビジネス・オブジェクトは，システムで管理すべき情報を，その属性としてもっており，通常，永続オブジェクトとなる．

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・オブジェクトと共にアプリケーション・システムを構成しているクラス群である．アプリケーション・オブジェクトは，ドメイン・オブジェクトとコンピュータ・システムを整合させるためのオブジェクトと言うことができる．アプリケーション・オブジェクトは，ビジネス・ルールにもとづいてビジネス・オブジェクトを制御するための**コントロール・オブジェクト**，アプリケーション・システムとシステムの利用者との間の情報交換を仲介する**インタフェース・オブジェクト**，アプリケーション・システムとデータベースの間のスキーマ変換とデータ交換を仲介する**データアクセス・オブジェクト**からなる．

サービス・オブジェクト

サービス・オブジェクトは，再利用可能なオブジェクトやプログラムコードである．他のオブジェクトから再利用可能なオブジェクトを**ユーティリティ・オブジェクト**と呼ぶ．コンテナ・オブジェクトやクラス・ライブラリーなどがユーティリティ・オブジェクトに含まれる．また，直接再利用可能なソフトウェア・モジュールを**コンポーネント**と呼ぶ．アプリケーション・システムから呼び出される既存のシステムは，コンポーネントとなる．

4. BOAD 法のプロセスとモデル

BOAD 法では，アプリケーション・システムの開発プロセスを要求定義，オブジェクト分析，オブジェクト設計，システム設計，実装の五つのフェーズに分類する．このうち，オブジェクト設計フェーズとオブジェクト分析フェーズの作業を合わせてモデリングという．BOAD 法は，オブジェクト分析とオブジェクト設計のフェーズを支援するモデリング技法である．（図 2）

オブジェクト指向によるモデリングは，指定された作業を順番にこなして行くこれまでの手続き駆動型のプロセスとは異なり，問題領域に関するモデルを順次詳細化し，システム・モデルに変換するというモデル駆動型のプロセスを取る場合が多い．BOAD 法もモデル駆動型のプロセスを採用している．

要求定義フェーズでは，アプリケーション・システムへの要求を様々な視点から定義する．要求には機能に関する要求，性能に関する要求，品質に関する要求などがある．

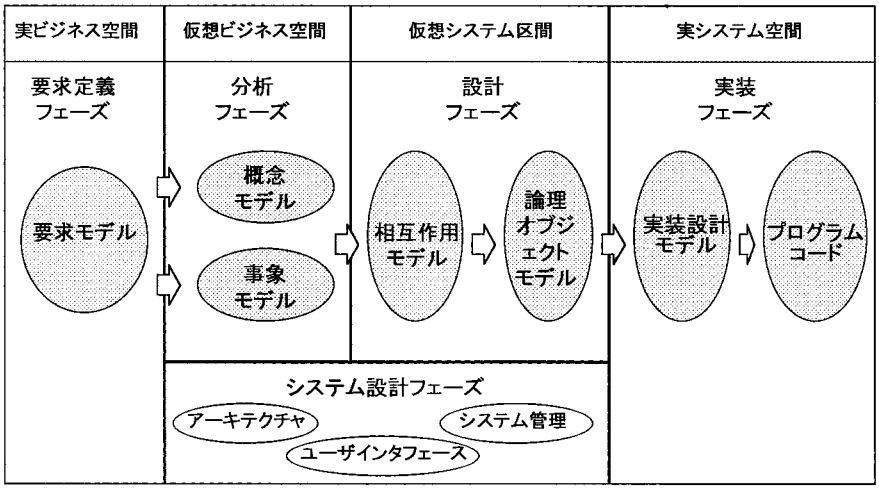

図 2　BOAD 法におけるプロセスとモデル

オブジェクト分析フェーズでは，要求定義にもとづいて，実世界の問題領域を抽象化し，その構成要素である実体や事象にもとづいて，分析モデルを作成する．分析モデルは，システムの外部仕様である．

実世界は，それを観察している人の感覚器官を通して認識され，概念化されて脳の記憶領域に記憶される．これを認知モデルと呼ぶ．オブジェクト分析では，この認知モデルを空間的な視点で捉えた概念モデルと，時間的な視点で捉えた事象モデルで表現する．これが分析モデルである．分析モデルを構成しているオブジェクトは，問題領域に存在している実体の概念，すなわちドメイン・オブジェクトである．

オブジェクト設計フェーズでは，概念モデルと事象モデルを統合してコンピュータ・システム上に問題領域を再構築する．これを設計モデルという．設計モデルは，システムの内部仕様である．

実世界の二つのビューである概念モデルと事象モデルをコンピュータ・システム上に再統合するのが相互作用モデルである．相互作用モデルには，ドメイン・オブジェクトと共に，参照モデルで定義されているアプリケーション・オブジェクトも含める．相互作用モデルは，システムを構成するこれらのオブジェクト間の相互作用を定義する．さらに，それぞれのオブジェクト間の相互作用をもとに各クラスの操作を定義することによって，論理モデルを作成する．設計モデルを構成しているオブジェクトはソフトウェア・モジュールとしてのオブジェクトである．

実装環境が決定したら，論理モデルを実装言語の仕様にあわせて実装モデルに変換する．この実装モデルをもとにプログラムの作成をおこなう．もしプロトタイピングが可能なら，実装設計を行わず，論理モデルから直接プロトタイプ・システムを作成しても良い．このように，実際に使用する実装環境上で実行可能なシステムを作成するプロセスを**実装フェーズ**という．

システム開発には，オブジェクト・モデリング以外にもアーキテクチャ設計やユーザインタフェース設計，システム管理設計などを行う必要がある．このようなプロセ

スを**システム設計フェーズ**と呼ぶ．

BOAD 法のモデルとその表記法は，以下の通りである．

概念モデル

概念モデルは，分析モデルの一つで，認知モデルのなかの空間モデルを明示化したモデルである．すなわち，概念モデルは問題領域を空間軸に沿って抽象化したモデルであり，問題領域をクラス構造によって表現した機械論的なモデルである．概念モデルは UML のクラス図によって表記する．

事象モデル

事象モデルは，認知モデルのなかの時間モデルを明示化した分析モデルである．すなわち，問題領域を時間軸に沿って法則化したモデルであり，問題領域の中で継起する事象の因果関係をシナリオによって表現した目的論的なモデルである．事象モデルは UML のユースケース図とシナリオによって表記する．

相互作用モデル

相互作用モデルは，空間と時間という二つの視点からモデル化された概念モデルと事象モデルをコンピュータ・システム上で一つに統合し，それぞれのオブジェクトの振る舞いを定義するためのモデルである．相互作用モデルの設計にあたってアプリケーション・オブジェクトが導入される．相互作用モデルは UML のシーケンス図によって表記する．

論理モデル

論理モデルは，アプリケーション・システムを構成するクラスの論理的な定義を与えるモデルである．論理モデルは UML のクラス図によって表記する．

5. モデリング

BOAD 法のモデリング作業の手順について，簡単に紹介する．ただし，システム設計やモデルの評価などについては省略してある．BOAD 法の詳細については，参考文献[5]がある．

5.1 分析モデル作成

モデリングの最初の作業は，問題領域の中からシステム化領域を定義することである．システム化領域の定義とは，要求定義をもとに，システムが提供すべきビジネス機能の範囲を明確にすることである．ビジネス機能と一口に言っても，それは観る人の立場の違いによってさまざまな様相を呈するに違いない．I. Jacobson ら[6]の提唱するユースケース分析は，その立場をシステムの利用者に限定することによって，統一的な視点からのシステムの機能の定義を可能にするものである．システムの利用者は，問題領域とコンピュータ・システムの接点に位置し，コンピュータを通してビジネス規則を観るという視点を提供してくれる．BOAD 法のシステム化領域の定義では，このユースケース分析法を利用する．成果物は，ユースケース図によって表記する．

図 3 において，人型のアイコンで表してある「顧客」や「メーカ」はアクタと呼ばれ，システムの利用者のそのものではなく，利用者の役割を表している抽象化したも

のである．利用者だけではなく，そのシステムを利用する外部システムもアクタとなる．システム化領域の定義は，まずアクタの識別から始まる．アクタが識別できたら，それぞれのアクタごとにシステムに要求する機能を定義してゆく．これがユースケースである．四角の箱がシステムで，その中の楕円形で表してあるのがユースケースである．アクタとそれに関連したユースケースを関連づける．ここでは，アクタ「顧客」からは「受注」と「在庫確認」というユースケースが，また，アクタ「メーカ」からは「発注」と「在庫確認」というユースケースが利用されることを表している．

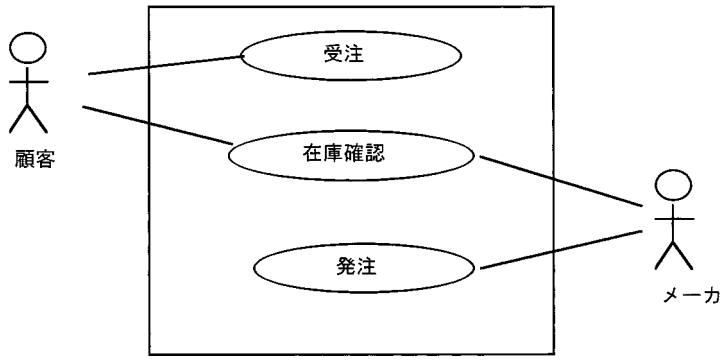

図 3　システム化領域の定義

システム化領域が確定したら，問題領域の中からオブジェクトの抽出を行い，問題領域をクラス図によって表記する．問題領域をクラスとクラス間の関係で表したものこれを概念モデルと呼ぶ．概念モデルは，問題領域を構成しているクラスの内部構造（名前，属性，サービス）と外部構造（汎化関係，構成関係，関連）を定義する．概念モデルは，問題領域を空間軸に沿って抽象化したモデルである．ただし，問題領域そのものをモデル化することはできない．モデル化できるのは，その問題領域を理解している当事者達のイメージ，すなわち認知モデルである．このモデルを概念モデルと呼ぶ理由がそこにある．同一の問題領域であっても，それを認識する人の立場や役割が異なれば，異なったモデルがうまれることになる．また，使用する方法論が異なれば，異なったモデルが作成されるに違いない．データ中心のオブジェクトや機能中心のオブジェクトのような議論が生まれる所以である．だから，問題領域のモデルには，妥当なモデルというのはあっても，正しいモデルというのは無い．しかし，同一プロジェクト内のモデルに関して言えば，後工程の作業者がそのモデルの意図を容易に理解できるように，モデリングの視点が統一していることが重要である．Wirfs-Brock ら[7]の責任駆動法はモデルの視点を統一するのに有益であり，CRC カードは問題領域の専門家たちとの作業に有効である．責任駆動法というのは，個々のオブジェクトを，問題領域の中における責任（責務）によって定義しようとする方法で，各オブジェクトを「何を知っているか」,「何ができるか」という視点で捉えようとするものである．クラスの識別というオブジェクト指向システム開発の中心的課題に決定

的な方法が無い現在，責任駆動法は最も有効な技法の一つである．概念モデルは UML のクラス図で表記する．

図 4 では，四角の箱での中に示されているのがクラスである．クラス「顧客」は，「個人顧客」と「企業顧客」という二つのサブクラスの汎化クラスとして定義され，クラス「受注」と「注文」という関連で連結している．また，クラス「受注」は，「受注明細」というクラスから構成されている．「顧客」と「受注」は，「注文」という関連で結ばれている．この図では，各クラスの属性やサービスのような内部構造は煩雑なので示されていない．

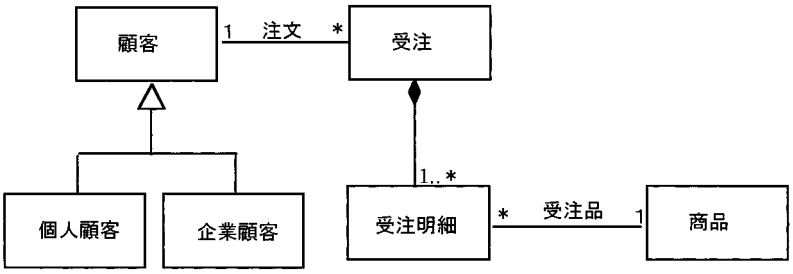

図 4　概念モデル

もう一つの分析モデルは，事象モデルである．ビジネス活動の中で継起する事象の連鎖を時系列に記述する．すなわち，事象モデルは問題領域を時間軸に沿って抽象化したモデルである．このような問題を記述する方法として UML には幾つかの動的モデルが定義されているが，BOAD 法では，自然言語によるシナリオによって問題を記述することにしている．それは，実世界のビジネス活動が持っている多様な意味を，できるだけ自由に記述したいからである．形式的な記述は厳密である反面，捨象される情報も多い．どちらの方式を採用するかは，考え方の問題である．シナリオは，ユースケースの単位毎に記述する．シナリオが記述できたら，各シナリオをもとに，共有サブシナリオや拡張サブシナリオを分離独立することによって構造化を図る．共有サブシナリオとは，複数のシナリオが共通に持っているサブシナリオのことであり，拡張サブシナリオとは，シナリオの主系列に対し副次的な系列のことである．これらのサブシナリオを，新たなユースケースとする．この構造は，ユースケース図によって表記する．

図 5 では，「受注」というユースケースと「発注」というユースケースから「在庫更新」という共有ユースケースが分離独立されている．また，「受注残」というユースケースは，ユースケース「受注」の拡張シナリオである．

5.2　設計モデル作成

設計は，問題領域を空間軸と時間軸に沿って抽象化した概念モデルと，時間軸に沿って法則化した事象モデルという二つのモデルを，コンピュータ・システム上で再統一する．このとき，問題領域のモデルである分析モデル（ドメイン・オブジェクト）をコンピュータ・システム上で実行するために必要な機構を追加する．これをアプリケーション・オブジェクトと呼ぶ．アプリケーション・オブジェクトは，参照モデル

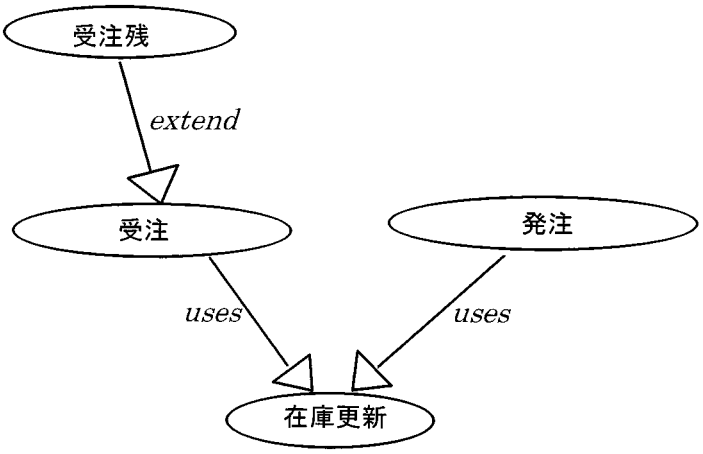

図 5　ユースケース図

に基づいて設定する．ここで設定するのは，インタフェース・オブジェクトとコントロール・オブジェクトである．システムを構成するのに必要なオブジェクトが出そろったらこれらのオブジェクトと前述のビジネス・シナリオを統合することによって相互作用モデルを作成する．相互作用モデルは，ビジネス・シナリオを実現するためのオブジェクト間の相互作用を設計するものである．そしてオブジェクト間の相互作用は，メッセージ・パッシングに変換される．相互作用モデルは，シーケンス図によって表記する．

図 6 で，「I/F」はインタフェース・オブジェクトを，「control」はコントロール・オブジェクトを表している．その他のオブジェクトは，ドメイン・オブジェクトである．左側には，ビジネス・シナリオが，そのシナリオ番号によって示されている．アクタからシステムに入力された刺激は，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に変換され，コントロール・オブジェクトに渡される．コントロール・オブジェクトは，ビジネス・シナリオにもとづいてドメイン・オブジェクトをス

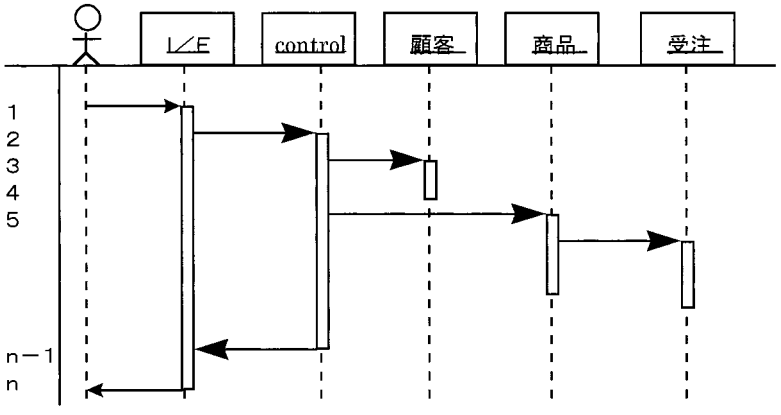

図 6　相互作用モデル

ケジュールする．メッセージを受け取った各ドメイン・オブジェクトは，自身の責任にもとづいて必要な情報処理を実施する．

相互作用モデルが完成したら，それをもとに論理モデルを設計する．すなわち，相互作用モデルに描かれたオブジェクト間のメッセージ・パッシングと事象モデルのシナリオをもとに各クラスの操作を設定する．こうして設定された操作のシグニチャとアルゴリズムや，属性のデータ構造を定義したものが論理モデルである．論理モデルは，クラス図によって表記する．

6. おわりに

モデリングという作業は創造的な作業である．知識を習得したからできるというものではない．何よりも重要なのは，与えられた問題を自分のものとしてこなせるかどうかであり，そうした技術は多くの実戦経験の中で養われるに違いない．しかし一方で，モデリング作業の標準化を図ったり，作成されたモデルの品質を高めるためには何らかの規準が必要となる．方法論とは，そのようなときに役立つものである．モデラーの持つ創造力をいかに高めることができるかが，方法論の究極の目標であるに違いない．BOAD 法の目標もそこにある．

参考文献

［1］ B. Meyer, “ Object-Oriented Software Construction ” Prentice-Hall, 1988（邦訳：二木厚吉監訳「オブジェクト指向入門」, アスキー出版, 1990）.

［2］ 妻木俊彦, 岩田裕道, 森澤好臣：「再利用性と拡張性のためのソフトウェア構築技法と評価指標について」, 情報処理学会「オブジェクト指向‘ 98 シンポジウム」論文集, 1998, pp 112-115.

［3］ 妻木俊彦, 岩田裕道, 森澤好臣：「オブジェクト指向モデリング方法論 BOAD 法」, 情報処理学会研究報告 98-SE-118, 1998, pp 71-78.

［4］ http : //www.rational.com/uml/index.shtml

［5］ 妻木俊彦, 岩田裕道：「ビジネス・アプリケーションのためのオブジェクト指向モデリング」, 日刊工業新聞社, 1999.

［6］ Jacobson, I., etc. “ Object-Oriented Software Engineering ”. Addison-Wesley Publishing Company. 1992.（邦訳；「オブジェクト指向ソフトウェア工学 OOSE」, トッパン）

［7］ WirfsBrock, R., etc.“ Designing Object-Oriented Software ”. Prentice Hall. 1990.

執筆者紹介 妻 木 俊 彦（Toshihiko Tsumaki）

1970 年東北大学理学部卒業．同年日本ユニシス(株)入社．大型計算機の OS，ネットワーク・アーキテクチャ，人工知能システム，オブジェクト指向システムなどの企画・開発に従事．現在，情報技術部技術研究開発室所属．ACM 会員．