ソフトウェア工学 95－7
（1993. 11. 30）

オブジェクト指向概念の発生学的定義に基づくソフトウエア設計技法

中所武司
明治大学理工学部情報科学科
email : chusho@cs.meiji.ac.jp

最近、オブジェクト指向技術を用いたソフトウエア開発技法が注目され、開発の容易化やソフトウエア再利用の促進が期待されているが、オブジェクトの設計方法が重要課題である。既に提案されている技法は、データモデルに基づくオブジェクトの静的な定義が中心のため、ビジネス分野の定形業務には向いているが、今後の分散コンピーティングやエンドユーザコンピューティングの発展に対応した非定形業務の開発には必ずしも適していない。

本報告では、これらの新分野を対象としたオブジェクト指向設計技法を提案する。まず、データモデルよりも分散協調型の計算モデルを重視し、オブジェクト指向の基本的な概念を発生学的観点から定義する。次に、この定義に基づき、ミクロモデルよりもマクロモデル、静的構造よりも動的ふるまいを重視した設計プロセスを提案する。

Software Design Method Based on Genetic Definitions of Object-Oriented Concepts

Takeshi Chusho
Meiji University, Department of Computer Science

Most of conventional object-oriented design techniques are suitable for application software of business information systems such as database-centered systems, but they are not suitable for application software of office systems in the new trend of end-user computing or distributed computing.

This paper presents new object-oriented design process for such new fields. Firstly essential object-oriented concepts are genetically defined from the viewpoint of a computation model, not of a data model. Design process based on these definitions are provided while paying attention to a macro model rather than a micro model and to dynamic behaviour rather than static structure.

1．はじめに

最近、ソフトウエア開発技法の１つとして、オブジェクト指向設計技法が注目されている。その背景には、ＣＡＳＥの発展と方法論のパラダイムシフトという２つの技術トレンドがある。

近年、上流工程の重要性が再認識され、それまでプログラミング環境が中心だった開発環境は、上流から下流、さらには保守工程まで一貫して支援する統合ＣＡＳＥへと発展してきた。ここでの重要な課題は、情報の共有やユーザインタフェースの統一に加えて、統一的な開発方法論による統合化[2]である。

しかし、従来の方法論は、要求分析技法、設計技法、プログラミング技法が個別に開発されてきたため、相互の接続のところにセマンティクギャップがあり、障害となっていた。この解決のために注目されているのがオブジェクト指向の技術である。

最近では、ソフトウエアの種々の分野でオブジェクト指向のソフトウエアアーキテクチャ[6]が採用されている。即ち、ワークステーションのオープン化に呼応してソフトウエアのアーキテクチャを階層化するとともに、各階層を統一的なインタフェースを有する部品オブジェクトで構成する方式である。その結果、開発者にとっては部品化と再利用、保守者にとっては拡張性と移行性、利用者にとっては操作性と統一性の利点をもたらしている。

オブジェクト指向設計技法は、これらのトレンドを背景に１９８０年代から研究が活発化し、従来の構造化技法と対比するかたちで発展してきた。既に多くの技法が提案されているが、データモデルの概念やデータ抽象化の概念を基本にしたものが多く、設計の初期の段階でのオブジェクトの静的な定義を重視している。このような開発手順は、既に基幹データベースを有するビジネス分野の定形業務システムの構築には向いているが、今後の分散コンピューティングやエンドユーザコンピューティング[7,8]の発展に対応した非定形業務の開発には必ずしも適していない。

本報告では、これらの問題を解決するために、従来のオブジェクト設計技法とは異なり、データモデルよりも計算モデル、ミクロモデルよりもマクロモデル、オブジェクトの静的構造よりも動的ふるまいを重視した設計技法を提案する。本技法は、著者らの過去のオブジェクト指向言語[3,4]およびオブジェクト指向言語をベースにしたマルチパラダイム型言語[5]の研究開発の経験に基づくものである。２.で従来方式の課題、３.で本技法の基礎となるオブジェクト指向概念の発生学的定義、４.でそれに基づく設計プロセス、５.で記述実験例を述べ、６.で結果の検討を行う。

2．オブジェクト指向設計の課題

2．1 オブジェクト指向概念の定義

オブジェクト指向概念の定義は必ずしも明確ではないが、actorモデル[13]などのメッセージ交換による分散協調型計算モデル、Simula67[10]からSmalltalk80[12]につながるプログラミング言語概念およびＥＲモデルなどのデータモデルがベースになっている。既存のオブジェクト指向設計技法は、G.Booch[1], P.Coad &E.Yourdon[9], S.Shlaer & S.J. Mellor[17], J.Rumbaugh[16]等、などをはじめとしてデータモデルに重きを置いたものが多い。

本報告では、システム全体の動的ふるまいに着目する立場から、計算モデルとしての位置付けを重視し、分散協調型モデル、抽象データ型、クラスからのインスタンス生成、継承機能付きクラス階層をオブジェクト指向概念の主要なものとするが、詳細は後述する。

2．2 複合モデル化と設計プロセス

設計技法をモデリング技法とみた場合、従来方式は３種類のモデルを組み合わせたマルチパラダイム型のものが多い。S.Shlaerらの情報モデル、状態モデル、プロセスモデル、J.Rumbaughらのオブジェクトモデル、動的モデル、機能モデルなどがその例である。これらの技法に共通する特徴は、設計プロセスの初期の段階でオブジェクトおよびオブジェクト間関係の定義を重視することである。そのため、とくにオブジェクト間関係のビジュアルな表記法が豊富に導入されている。

これらのボトムアップ的アプローチは、既に開発運用実績を有する従来の基幹データベースを中心としたビジネスシステムの再構築のような場合には適用可能である。しかし、システム全体のマクロレベルの動的ふるまいの設計法があいまいであることや３つのモデルの統合が不明確などの欠点[11,15]があるため、今後増加していくと思われ

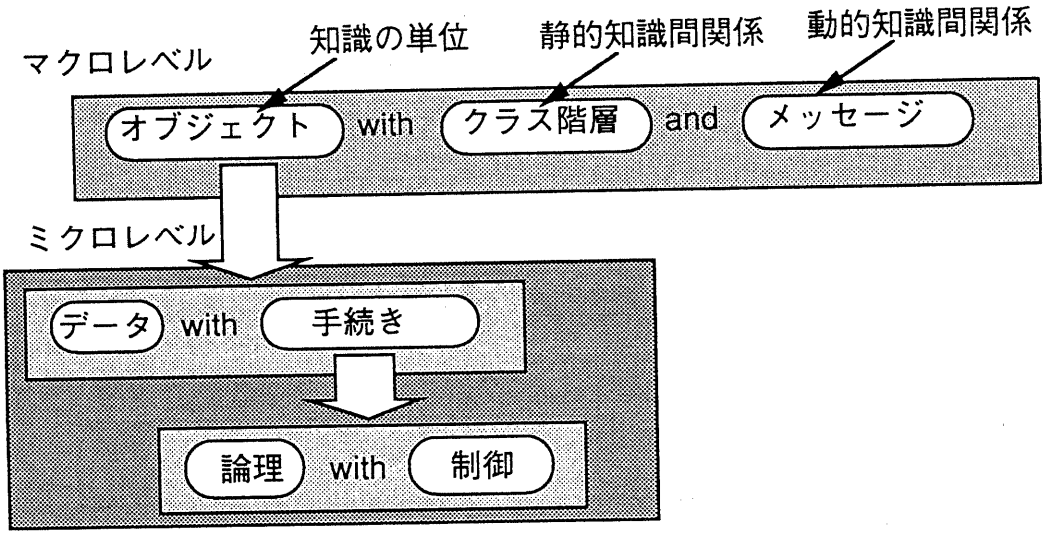

図1　オブジェクト指向言語設計の観点での2階層モデルの図式

るエンドユーザコンピューティングに近い非定形業務の新規開発には適さない。

このような分野では、何をオブジェクトにするかを決定するためには、データモデルよりも計算モデルを重視し、モデリングの初期の段階で、従来の技法であまり明確に示されていないオブジェクト群の動的なふるまいを記述することが重要である。本報告では、図1に示すように、著者らがオブジェクト指向を基本としたマルチパラダイム型言語[3]の開発時に提案した2階層モデルをベースに、ミクロモデルよりもマクロモデルを、静的構造よりも動的ふるまいを優先する設計プロセスについて述べる。

2．3 問題領域の多様性

ソフトウエア開発の問題領域は多様であり、汎用的な設計プロセスは難しい。本節では、本報告の主たる対象を明確にしておくが、それ以外の分野に適さないということではない。

（1）問題領域依存性

ビジネス分野の基幹業務向きのデータモデル中心の設計法、エンジニアリング分野の状態遷移モデル中心の設計法に対し、本報告では、OA分野の非定形業務を対象とした分散協調型モデルに基づく設計法を提案する。

（2）大規模 vs. 小規模

オブジェクト指向設計技法における大規模対応機能としては、P.Coadらのサブジェクトの概念による構造的グループ化、S.Shlaerらのサブシステムの概念による機能分割、P.Jalote[14]のネスト構造による段階的詳細化、あるいはG.Boochのis-a階層とpart-of階層を組み合わせた直交階層化などが提案されている。大規模な問題あるいは複雑な問題を扱う常套手段は「分割統治」の原則に従ってグループ化、階層化、ネスト構造化などによりサブ問題に分割することである。本報告の対象分野は比較的小規模のものが多いが、大規模化に対しては同様の手段が必要になる。

（3）再開発 vs. 新規開発

開発方法論が異なっても既存の類似システムが存在する場合は、データベースやファイルの構造あるいはユーザインタフェースの操作などについて、ある程度の見通しが得られるので、クラスオブジェクトの設計やクラスオブジェクト間の関係の定義を初期の段階で行うことが可能である。しかし、本報告の対象分野は、一般に前例のない新規開発の場合が多く、トップダウンに設計する必要がある。

（4）受注ソフト vs. エンドユーザコンピューティング

受注ソフトウエアは一般に大規模のものが多く、多人数開発になるので、ウォータフォールモデル

あるいはその改良方式を用い、各工程で仕様を検証しながら開発を進めるフェーズドアプローチをとる。それに対し、本報告では、エンドユーザコンピューティングに近い分野を対象とするため、まず核になる部分を試作し、それを改良しながら実用システムに仕立てあげるプロトタイピングアプローチをとる。

3．オブジェクト指向概念の発生学的定義

オブジェクト指向概念の基本的な要件として、計算モデルの観点から以下の4項目をとりあげる。

- ●分散協調型計算モデル
- ●データ抽象化機能
- ●クラスからのインスタンス生成機能
- ●クラスの階層化と継承機能

これらのオブジェクト指向概念をより厳密に規定するために、著者らが以前に言語設計時に用いたオブジェクト指向言語モデルの形態形成過程[3]に基づく発生学的定義を以下に示す。即ち、ある概念の存在が前提となって別の概念が導かれるとき、前者が後者よりも先に定義される。なお、以下では特に断わらないかぎりオブジェクトという表現はインスタンスを意味する。

（1）メソッドとメッセージ送信

複数個の自律的機能を持つオブジェクトがお互いにメッセージをやり取りしながら協調して問題解決にあたるという計算モデルの中で、個々のオブジェクトの機能（外部仕様）は、メソッドの集合とする。各メソッドはそれを指定したメッセージの受信によって起動され、所定の機能を果たす。（分散協調型計算モデルの形成）

（2）データ

個々のオブジェクトは、オブジェクトの状態を保持するために、必要ならばデータ（変数）を有する。これらのデータへのアクセスは、それらと同じオブジェクト内のメソッドからのみ可能とする。（データ抽象化機能の形成）

（3）クラスとインスタンス

メソッドが同じで、データの内容が異なるオブジェクトを効率よく作成するために、もとになるオブジェクトを型として、その実体を複数個生成（複製）可能とする。このとき、型となるものをクラス、生成されたものをインスタンスと呼ぶ。（インスタンス生成機能の形成）

（4）クラスメソッドとインスタンスメソッド

メソッドにはクラス用のものとインスタンス用のものがあるので、それぞれクラスメソッドとインスタンスメソッドに分離する。

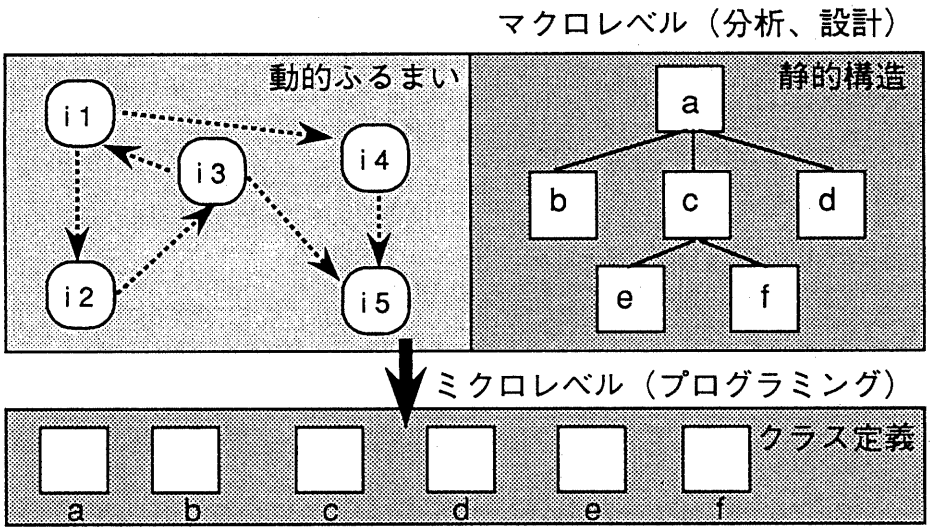

図2　オブジェクト指向設計の観点での2階層モデルの概念図

（5）クラス変数とインスタンス変数

クラスメソッドの操作対象となるデータとインスタンスメソッドの操作対象となるデータを分離し、クラス変数およびインスタンス変数とする。

（6）クラス階層と継承機能

メソッドの集合が少しづつ異なるオブジェクトを効率よく作成するために、クラス階層を導入する。そして、下位のクラスは上位のクラスの性質、すなわちメソッドと関連データを継承できるものとする。（クラス階層と継承機能の形成）

このような言語モデルの概念形成過程において、前のものほどオブジェクト指向概念として本質的な機能と考えられる。ただし、一般には、このような機能に基づく利点はユーザ視点で決まるものであり、その実現のために必要な機能が備わっていれば、すべての機能がなくても「オブジェクト指向」という用語が用いられている。

4．オブジェクト指向設計のプロセス

4．1 基本方針

コンピュータで取り扱う問題に関連した実世界の概念的対象物をそのままプログラムの基本単位として表現できるオブジェクト指向概念の特長を活かすためには、発生学的定義で規定した各構成要素を、図2の2階層モデルの枠組みの中でその定義順に設計していくのが最も自然と思われる。このような方式の設計プロセスは従来のオブジェクト指向設計技法と次のような点で異なっている。

●マクロモデルをミクロモデルよりも優先。
●計算モデルをデータモデルよりも優先。
●動的ふるまいを静的構造よりも優先。
●インスタンス定義をクラス定義よりも優先。

本方式の主なプロセスは次のようなものである。

(1) 実世界の問題領域のモデル化
(2) システム化する解領域のオブジェクト設計
(3) オブジェクトの実現
(4) 実行、検証、改良

以下では、プログラミング言語に依存しないマクロレベルの(1)と(2)の具体的な手順について説明する。プログラミング言語に依存するミクロレベルの(3)と(4)は具体例の中で簡単に述べる。

4．2 実世界の問題領域のモデル化

まずはじめに実世界の問題領域を分析し、メッセージ送信による分散協調型計算モデルの視点から、マクロなレベルでの動的ふるまいの概要を描く。それと同時に「もの」の存在のシンボルとしてのインスタンスオブジェクトおよび「もの」の役割（機能）のシンボルとしてのメソッドを抽出する。ここで役割とは「もの」がシステムの中に存在（実存）する意味を与えるものである。問題領域に存在するものがすべて解領域のオブジェクトになるわけではない。従って、オブジェクト指向設計技法は、よくいわれるようにデータ指向に近く、構造化分析や構造化設計のようなプロセス指向または機能指向ではないというよりも、両者を対立的に超越（止揚）したものである。

4．3 解領域のオブジェクト設計

（1）メソッドの外部仕様の明確化

まずシステム化する解領域を明確にするために、外界との境界となるインスタンスのメソッドの外部仕様を明確にする。次にシステム内部でやりとりするメッセージを受け取るインスタンスのメソッドの外部仕様を明確にする。

（2）内部データの導入

各々のインスタンスの内部データを導入し、そのデータ構造を詳細化する。

（3）メソッドの内部仕様の詳細化

各々のメソッドの処理内容の実現方式をきめる。前記（2）のデータ構造の詳細化とこのメソッドの詳細化のレベルは抽象度が同じになるようにする。

（4）クラス定義

データの値のみが異なる複数のインスタンスは同一クラスとする。この時、必要に応じてクラスメソッドとインスタンスメソッドを分離する。これにあわせてクラス変数とインスタンス変数を分離する。

（5）クラス階層の導入

機能が類似のクラスはそれらに共通の上位クラスを導入し、クラスの階層を形成すると共に、継承機能によってその共通部分を利用する。

ここで示したプロセスは逐次的手順であるが、

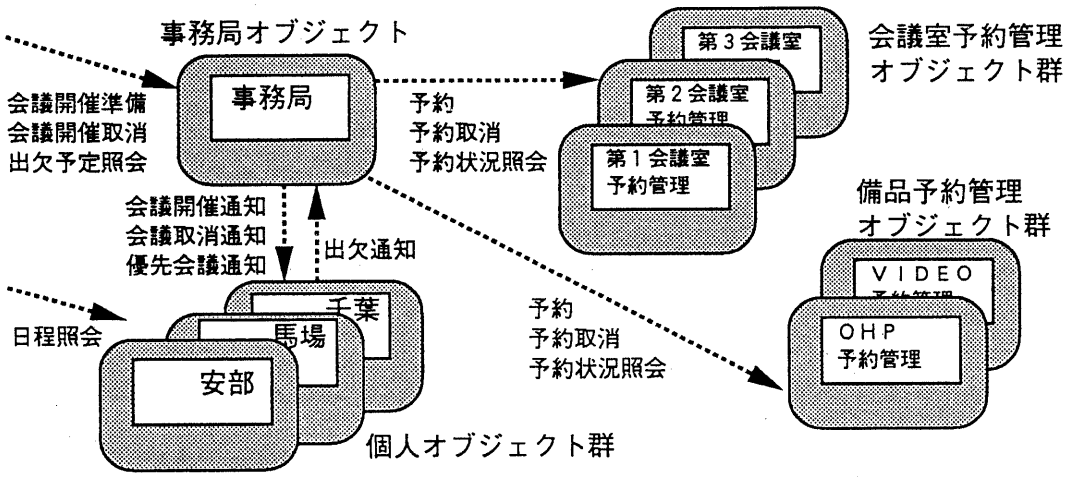

図３　分散協調型モデル（会議開催事務処理の例）

現実には個別の問題対応に次のような例外がありうる。

(1) 近接する複数の手順が並行する。例えば、内部データの構造とそれを操作するメソッドの内部仕様が同時に決定されていく。

(2) 一部分の手順が省略される。例えば、クラスメソッドを必要としなければ、クラスメソッドやクラス変数の定義は不要である。

(3) 後の手順が先行して実行される。例えば、マクロモデルの中の複数のインスタンスからの同一のクラスの導出やクラス階層の導出が容易に推測可能な場合は、クラス定義をインスタンス定義よりも先に行う。

５．記述実験による検証

５．１　例題の概要

本報告で提案するオブジェクト指向設計法の適用実験のための例題として、会議開催事務処理業務を取り上げる。この業務はこれまで実世界では人手作業で行なわれていたとする。そのアプリケーションソフトウエアの開発にオブジェクト指向概念を適用するとき、実世界とシステムの対応をとるために、最初に何をオブジェクトとするかが問題となる。その決定手順の例を以下に示す。なお、この会議開催事務処理システムの名前をＯＯＯ（Object-Oriented Office）としておく。

５．２　実世界の問題領域のモデル化

まずはじめに実世界の問題領域を分析し、分散協調型計算モデルの視点から、マクロなレベルでの動的ふるまいの概要を描く。会議開催事務処理は、図３に示すように、以下の４種類のオブジェクトを導入し、互いにメッセージをやり取りしながら協調して問題解決にあたるようにモデル化できる。図の角の丸い箱はオブジェクト、破線はメッセージ送信を示す。

■事務局オブジェクト

会議開催の要求メッセージを受けると、会議室の予約、ＯＨＰの予約、その会議のメンバへの会議開催案内などのメッセージを各々の担当オブジェクトへ送り、その後、メンバからの出欠通知のメッセージを受け取り、管理する。会議の中止や出欠の変更、出欠予定の状況報告などの処理もする。

■個人オブジェクト

分散コンピューティングの環境下でのグループウエアを想定し、個人毎のスケジュール管理をする。安倍、馬場、千葉の３氏のオブジェクトは会議開催案内のメッセージを受け取り、出欠通知のメッセージを送る。優先度の高い会議が後から入った場合は、先約の会議の出席を取り消す。スケジュールの状況報告も行う。

■会議室予約管理オブジェクト

各会議室毎に会議室予約のメッセージを受け取り、予約管理をする。予約取消処理や予約状況報

告も行う。

■備品予約管理オブジェクト

各備品毎に備品予約のメッセージを受け取り、予約管理をする。予約取消処理や予約状況報告も行う。

5．3 解領域のオブジェクト設計

（1）メソッドの外部仕様の明確化

まずシステム化する解領域を決定する。そのために、外界との境界となる事務局オブジェクトと個人オブジェクトのメソッドの外部仕様を明確にする。次にシステム内部で閉じた機能を果たす会議室予約管理オブジェクトと備品予約管理オブジェクトのメソッドの外部仕様も明確にする。以下にメソッド名と引数を示す。

■事務局オブジェクト

・会議開催準備（会議名,月,日,開始,終了）
・会議開催取消（同上）
・出欠通知（出欠者名,出欠,会議名）
・出欠予定照会（会議名）

■個人オブジェクト群

・会議開催通知（会議名,月,日,開始,終了）
・会議取消通知（同上）
・優先会議通知（同上）
・日程照会（月、日）

■会議室予約管理オブジェクト群

・予約（会議名,月,日,開始,終了）
・予約取消（同上）
・予約状況照会（月,日）

■備品予約管理オブジェクト群

・予約（会議名,月,日,開始,終了）
・予約取消（同上）
・予約状況照会（月,日）

（2）内部データの導入

各々のオブジェクトの内部データとして、事務局オブジェクトに会議管理表、個人オブジェクトには手帳に相当する日程表、会議室予約管理オブジェクトと備品予約管理オブジェクトには予約ノートに対応する予約管理表を導入し、そのデータ構造を詳細化する。

（3）メソッドの内部仕様の詳細化

各々のメソッドの処理内容の実現方式をきめる。

（4）クラス定義

事務局オブジェクトに関してはそれだけが生成されるクラスを定義する。個人オブジェクト群は機能が同じなので同一のクラスとする。会議室予約オブジェクト群と備品予約オブジェクト群も機能が同じなので同一のクラスとする。

（5）クラス階層の導入

個人オブジェクトのクラスは、優先会議通知メソッド以外は予約管理クラスと機能が同じなのでその子クラスとし、優先会議通知メソッドのみを定義する。

5．4 オブジェクトの実現

この例では、オブジェクトの実現はプログラミング言語C＋＋を用いて行った。このミクロレベルのプログラミング段階でさらに以下のような幾つかのオブジェクトが導入されたが、詳細は紙面の都合で省略する。

(1) 抽象データ型オブジェクトの導入

会議室予約管理表、備品予約管理表、日程表、会議管理表。

(2) インスタンス名の管理オブジェクトの導入

6．結果の検討

マクロレベルでの動的なシステムのふるまいを示す分散協調型計算モデルを重視したオブジェクト指向設計法を提案し、その適用例を示した。その結果について2.3の課題の観点から考察する。

（1）マクロモデル優先の設計プロセス

本方式の主対象であるＯＡ分野の非定形業務の典型的な例を取り上げ、マクロモデルの構築の重要性を示した。特に最初にシステム全体の動的ふるまいを明確にすることが、適切なオブジェクトの抽出に不可欠である。

例えば、従来のデータモデルをベースにしたオブジェクト指向設計技法のように、最初の段階で、実世界（問題領域）に存在するものを「会議室があるから」、「黒板があるから」と次々とオブジェクトにする方法でうまくいかないことは明白であろう。データに注目して「会議室予約ノート」をカプセル化するという方法もプログラミング段階でのデータ抽象化の手法であり、メソッドがデータ操作手続きとして設定されてしまう。

本方式では、システム全体の動的ふるまいの中でのものの役割（機能）に注目する。例題では、

予約管理という役割から予約、取消、照会というメソッドを抽出することにより、最終的に会議室予約管理オブジェクトと備品予約管理オブジェクトが同一のクラスから生成可能であることを導いた。さらに個人オブジェクトがその予約管理クラスの特殊化によって得られることも導いた。このようなクラス階層の導出はデータのカプセル化の観点では難しい。

（2）プロトタイピング方式による拡張性

新規開発が主体の非定形業務は、最初に要求仕様を厳密に規定できないため、プロトタイピング方式の開発にならざるを得ない。この場合、変更内容に応じて修正範囲はいろいろであるが、総じて修正個所はわかりやすい。

（3）オブジェクト間の関係の記述

本例題では、オブジェクト間の関係に関しては、通常のis-aおよびhas-aの関係のほかに、メッセ-ジの送受信の関係があり、分散協調型モデルの中に示した。

大規模なシステム開発では、クラスの種類が多くなり、種々のクラス間関係が生じるため、既存のオブジェクト指向設計法で提案されているような記法が役に立つと思われる。

7．おわりに

コンピュータで取り扱う問題に関連した実世界の概念的対象物をそのままプログラムの基本単位として表現できるオブジェクト指向概念の特長を活かすために、そのオブジェクト指向概念を発生学的に定義し、そこで規定された各構成要素をその定義順に設計していく方式を提案した。

本方式の主対象であるＯＡ分野の非定形業務の典型的な例を取り上げ、適用実験を通じてマクロモデルの構築の重要性を示した。特に最初にシステム全体の動的ふるまいを明確にすることが、適切なオブジェクトの抽出に不可欠であることを明らかにした。

参考文献

1) Booch,G.:Object-Oriented Design with Applications, Benjamin/Cummings(1991).

2) Chusho,T.,Watanabe,T. and Hayashi,T.:A Language -Adaptive Programming Environment based on a Program Analyzer and a Structure Editor,Proc. the 9th World Computer Congress IFIP'83, 621-626 (1983).

3) 中所,芳賀:オブジェクト指向型言語と論理型言語の融合方式に関する考察,オブジェクト指向,共立出版,pp.133-146(1985).

4) Chusho,T. and Haga,H.:A Multilingual Modular Programming System for Describing Knowledge Information Processing Systems,Proc. the 10th World Computer Congress IFIP'86,pp.903-908(1986).

5) 中所,増位,芳賀,吉浦:マルチパラダイム型言語における対立概念の融合方式,人工知能学会誌, 4, 1,77-87(1989).

6) 中所武司:使いやすいソフトウエアと作りやすいソフトウエアーオブジェクト指向概念とその応用ー,電気学会雑誌,Vol.110,No.6,465-472(1990).

7) 中所武司:エンドユーザコンピューティングーソフトウエア危機回避のシナリオー,情報処理, Vol.32,No.8,pp.950-960(1991).

8) 中所武司：ソフトウェア危機とプログラミングパラダイム、啓学出版(1992).

9) Coad,P. and Yourdon,E.:Object-Oriented Design, Prentice Hall(1991).

10) Dahl,O., et al.:Hierarchical Program Structures, Structured Programming,Academic Press,175-220 (1972).

11) Fichman,R.G. and Kemerer,C.F.:Object-Oriented and Conventional Analysis and Design methodologies, IEEE Computer,Vol.25,No.10, pp.22-39(1992).

12) Goldberg,A., et al.:Smalltalk-80 the Language and its Implementation,Addison Wesley (1983).

13) Hewitt,C., et al.:Laws for Communicating Parallel Processes,Proc. IFIP'77,pp.987-992(1977).

14) Jalote,P.:Functional Refinement and Nested Objects for Object-Oriented Design,IEEE Trans. Softw. Eng., Vol.SE-15,No.3,pp.264-270(1989).

15) Monarchi,D.E. and Puhr,G.I.:A Research Typology for Object-Oriented Analysis and Design, Comm. ACM, Vol.35,No.9,pp.35-47(1992).

16) Rumbaugh,J.et al.:Object-Oriented Modeling and Design,Prentice Hall(1991).

17) Shaler,S., et al.:Object-Oriented Systems Analysis :Modeling the World in Data,Prentice Hall (1988).