記　　号　　処　　理　74－5
プログラミング－言語・基礎・実践－　17－5
（１９９４．　６．　３）

オブジェクトベース機構：
オブジェクト指向一貫モデリング過程論に基づくシミュレーションの実現

畠山正行　　金子　勇

茨城大学工学部情報工学科

本研究の目標は、対象世界（自然現象）に実在する”もの”に対応した自然なモデリングと”もの”のモデル化の単位モジュールが駆動するシミュレーションの実現である。その目標を達成するには対象世界を最も自然かつ首尾一貫したモデリンググパラダイムに従うモデリング過程を理論化した。そのモデリングパラダイムとしてはオブジェクト指向を用いた。現状の問題点は、実現モデリング段階でのパラダイムギャップである。これを解消するため、データと手続きを一体化し、”もの”オブジェクト単位でのアクセス及び起動・駆動・操作・制御しかできないような機構、即ち、オブジェクトベース機構をオブジェクト指向データベースを基盤にして考案・実現した。対象世界から再現シミュレーションまでを一貫したモデリング過程にすることで、”もの”のモデル化の単位モジュールが駆動するシミュレーションが実現した。

実現の対象世界の例として希薄気体流れのシミュレーションを行った。その結果、”物”のオブジェクトモデルをほぼ”物”と同等に取り扱うことが出来る方法がほぼ確立できたことを確認した。更には、手続き型のシミュレーションでは得られないような複雑な対象世界や動的に多様な変更が必要とされる対象世界に対して従来よりも遙かに簡単にシミュレーションが実現できることを確認した。

ObjectBased Mechanism:
A Mechanism for Simulations Based on the Object-Oriented Consistent Modeling Processes

Masayuki Hatakeyama, Isamu Kaneko

(Ibaraki University)

A Mechanism or a methodlogy for the simulations in the fields of the natural phenomena or the engineering structures is developed. The modeling processes from the original target world up to the realization simulation stage must consistently be processed. The programming stage, one of the modeling stage, is not consistent with the model of the target world. Therefore, to encapsule this modeling stage, the ObjectBased mechanism is originally developed. This mechanism is applied to the rarefied gas flow simulation problem, and then complex and flexible simulations are realized.

1．まえがき

計算機のそのもっとも計算機らしい特長の1つは、如何なる対象世界も人間の脳の認識モデルに発し、幾つものモデル化の段階を経て最終的には対象世界を表現するプログラムシステムに変換（モデリング）可能であり、出来の善し悪しはともかくとすれば、再現シミュレーションとして駆動させることが「出来る」ところにある、と我々は考えている。シミュレーションの対象世界は我々が認識できる世界なら一応なんでも再現シミュレーションとしてともかくも「可能」である。しかし、計算機によってあらゆる世界の現象を再現シミュレーションすることが出来る可能性を持っている、と同時に、我々は現状においては計算機の可能性を十分に引き出して活用するに至っていないとも考える。自然現象等を対象世界としての「再現的な模倣計算とその表現」においては、厳格に言えば、「シミュレーション（見かけもそれを実現している機構も対象世界のそれとsimilarである）」ではなく、「エミュレーション（見かけは同じであるが、それを実現する機構は対象世界のそれとは異なる）」であることが多いと言わざるを得ない。

1.1　一貫モデリング過程論の視点

我々はそれを一気に解決しようというものでは全くないが、ある視点からその1つ（しかもかなり大きな部分を占めると考えられる1つ）を体系的に取り出して、その取り出すときの過程から解決策を見いだすことを試みる。その視点とは、再現シミュレーションを実現するときに必ず行われなければならない対象世界のモデリング過程を改めて見直してみようという視点である。モデリング過程とは、第2章で詳細に述べるように人間が実世界に対して形成する認識モデルを対象世界をしてモデリング過程の原点とし、以下、大雑把に言えば、概念的な表現をする概念モデリング段階、論理的な構造等を厳密な用語と式等に依って表現する論理モデリング段階、（以上で分析モデリング段階終了）、分析結果を再構成し計算機上でのシステムとして駆動させるための設計を行う再構成モデリング段階、モデルの各要素をプログラミング言語を用いて実装し、シミュレーションを起動・駆動・制御・操作・持続的な格納・表現等を行う再現モデリング段階、等がある。モデリングにはこれらの大枠としてのモデリング段階の内部に詳細なモデル化をするモデリングステップ、その他複雑なモデリング過程があり得る。

我々の対象とする自然現象のモデリングとシミュレーション（エミュレーション？）において、注目すべきモデリングに関する「シミュレーション的でない点」とは、例えば、対象世界に実在する”もの”に対応するモデルの記述があるモデリング段階だけは全て他と異質な表現形態（パラダイムギャップ）を持つ場合である。即ち、実現モデリング段階では他のモデルリング段階のモデルと違って、対象世界の構造を表現する手段としてデータ構造と手続きに分離した手続き指向のモデリング以外に表現方式が無い。それ故に対象世界にもなく、他のモデリング段階にも存在しない異質なモデル記述を余儀なくさせられ、その駆動、即ちシミュレーション駆動においても対象世界の駆動様式や様子、及び、操作や制御の不便さ、シミュレーション条件の動的な変更の困難・煩雑さといった本来の対象世界にはない多くの不便な制限を強制させられている。

1.2　”物”に対応したモデリングとシミュレーション

このそもそもの原因は、対象世界に実在する”物”に対応した自然な”もの”へのモデリングと”もの”のモデル化の単位モジュールがその単位でシミュレーション駆動する機構と駆動方式が確立されていないからであると我々は考えた。なぜならば、対象世界の原モデルは”物”であり、それをモデリングパラダイムを一貫して変えないで必要十分条件を満たしつつモデリング変換をして行けば、実現モデリング段階においても”もの”のモデルがそのままの形で（データとプログラムに分解されずに、また手続き優先処理方式になる筈もなく）表現される筈である。となれば、実際の駆動においても”もの”に対応した様な駆動と制御と操作が実現されるはずであり、その実現シミュレーション駆動の様子はエンドユーザのいわゆるメ

ンタルイメージと計算機内のシミュレーション駆動のイメージが従来よりもかなり近いものになって来る筈である、と考えた。
具体的にどのようなモデリングにすべきかは対象世界から同じモデリングパラダイムに従って、首尾一貫したモデリング法で変換される過程の中から自然に導かれる。即ち、対象世界から始まり、実現シミュレーションモデリング段階に至るまでの全てのモデリング段階、モデリングステップにおいて一貫したモデリング過程を確立し、その中から、モデリングパラダイムや方法が異質なものを見いだして一貫したものに改革することで初めて、対象世界そのままの再現シミュレーションが実現する一筋の道が（当初は悪路でかつ細くはあっても）通じる可能性が出て来る、と考えた。

1.3 オブジェクト指向モデリング

次にモデリングパラダイムとして何を用いるべきかということであるが、我々は以下の理由からオブジェクト指向モデリングパラダイムを用いることとした。

1. 対象世界の”物”に対応したモデル化に適している。
2. 対象世界から再現駆動モデリングまでの一貫したモデリング過程が曲がりなりにも実現できる可能性を持つ実用上では唯１つのモデリングパラダイムであること。特にプログラミング言語のサポートがあるか否かが大きな要因である。
3. オブジェクト指向とは人間が意識的・無意識的に行っているモデリングのうちの少なくともある一定の水準以上のレベルでかつある一定の範囲以上の汎用性を持つモデリングを表現している、言い換えれば人間の思考モデリングのかなりの部分を Explicit に表現する際の「人間自体が行うモデリング」の総称に近いのではないかと思われること。

1.4 シミュレーション

さて、従来のシミュレーションにおいてはオブジェクト指向がモデリングパラダイムに意識的に採用され、モデリング過程を一貫してこれで表現及び変換し、プログラミング言語や駆動・制御・操作に至るまでオブジェクト指向的な実現をされたという例を我々は寡聞にして知らない。その理由は実現モデリングの異質なパラダイムでは一貫したモデリング方法を取ることは困難であったことに依るのではないかというのが我々の見解である。つまり、”物”から”モデル化されたもの”になり、その”モデル化されたもの”が実現モデリング段階においてモデル化された”もの”ものとは異質なデータ構造とプログラムの１セットに変換されざるを得無くなった。従って、モデル化された”もの”のもつ、従って、対象世界そのものが本来的に持つ特徴、例えば、極めて複雑な構造であるとか、駆動中の動的な変更であるとか、任意の観測すべき量やパターンの任意・臨機・即座の実現とか、がきわめて煩雑なプログラミングが必要であったり、言語の制限等で困難・不可能であったりした。

1.5 オブジェクトベース機構の考察

我々もこれを一気に根本的に解決仕様とはもちろんしていない。しかし、この点に関して当初はある程度の、そして段階的にかなりのレベルまで解決することを構想している。 まず最初のレベルとしては、実現モデリング段階においてデータと手続きを一体化するためのリンク機構の構築である。またオブジェクト生成段階の一部分ではあるが、出来上がってリンクを取られたデータ構造とプログラムを１セットにした機構を構築する。それは、”もの”オブジェクト単位での持続的格納・検索・取り出しを実現する。次に、”もの”オブジェクト単位でのアクセス及び起動・駆動・操作・制御しかできないような機構を提案してこれを実現する。即ち、これらを総称して、オブジェクトベース機構と呼ぶことにする。オブジェクトベースという概念は本質的にはオブジェクト指向と同義である [1] [2] [3] [4]。”物”のモデル化されたもの、＜データ構造＋メソッド＞を常に一体としたモデル、という意味を強調する意味で用いている。また、オブジェクト指向という用語の定義が広く、一定の合意された定義に基づかずに種々の意味に使われており、誤解され易いので我々は敢えてこの用語を用いている、という事情もある。

我々はこのオブジェクトベース機構をオブジェクト指向データベース管理システム(OODBMS)を基盤として構築することを試みる。OODBMSを基盤システムとして利用したのは

1．静的なオブジェクトの”オブジェクト単位”での管理（格納・検索・取り出し）が容易に実現できる可能性があったこと。
2．オブジェクト単位での起動を可能にする機能を副作用ながら備えていたこと。
3．シミュレーションの計算に関係する莫大でかつ多様なデータを必要に応じて持続的に格納・管理・利用できること。

というような特徴を備えているからである。但し、1．と2．においては我々の研究成果を含めてこそ言える特徴である。

モデルオブジェクトが実際にデータとメソッドに分かれて記述されるのは実装の部分、即ちプログラム記述の時のみである。そこでは対象世界の構造体全体はデータ構造で単純に置き換えられてしまっており、モデルの前段階で存在した本来の「構造体そのもの」は記述から消失している。従って、オブジェクト間の相互作用は置き換えられた（データ構造とメソッド）を構造体の代理物（モデル）として（当然同じ振舞いを生ずるであろうが）相互作用を仕掛けるということになる。従って、対象世界の静的な構造に対する全ての作用は、その構造体に対する（相互）作用を常時監視しておき、（相互）作用したと判断されたら、その構造が持つ（振舞いを起こす機能を持つ）機構を起動させ、あるいは反応すべき振舞いそのものを起動させ、従って、構造体を数値的に（データ的に）正確に表現するデータ構造に対して起動させたメソッドでもってアクセスさせ、その構造体が反応すべき振舞いを計算で求め、データの値を変更することで構造体が反応し、振舞い、ある静的な状態に戻ったと見なすわけである。これが（データ構造とメソッド）によって対象の動的な振舞いを実現している実態である。つまり構造体をデータ構造とメソッドの1セットで置き換えて（モデル化して）いるわけである。モデリング過程の全ての段階やステップ、即ち、認識モデル、概念モデル、論理モデル、再構成モデル、再現モデルの中で、モデルオブジェクトの表現形態が異なるのは再現モデルのみである。

1．6　オブジェクトベース機構の実現方針

本来、対象世界に物理的な実体としてあるのは物質から出来上がった構造（データ構造ではなく本物の構造）だけである。機構や振舞いを生じさせる（相互）作用の情報処理的実現としての手続きなどは対象世界の認識モデルには存在しない（書かれていない）。計算機のモデルには原理的に構造体そのものを記述するわけには行かない。従って、（データ構造とメソッド）の1セットで各オブジェクトを記述するのは止むを得ない処置（モデル化）であると言えよう。しかし、このモデル化は明らかにモデリングパラダイムの変更であり、従って、一貫したモデリング過程を実現するためにはこの実現モデリング過程におけるモデリングパラダイム過程の非一貫性を情報隠蔽する仕掛を作り、これによりその前後のモデリング過程（設計モデリング過程と駆動モデリング過程）の時とsimilarなモデリングパラダイムにする必要があることが分かる。

但し、この処置はモデリングパラダイムが一貫したものに変わったわけではなく、表面的な見かけだけが前後同じように見えるようになっただけであり、情報隠蔽してモデリングパラダイムが、従ってユーザのメンタルモデルが一貫したように見えるというだけのものであり、オブジェクト指向の抽象化のためのソフトウェア解釈の皮としては比較的薄いものである。また、実際にクラス定義を行うにはやはり、当然ながらデータ構造とメソッドを分けて書かなければならない。それはクラスオブジェクトのコーディング時の非オブジェクトベース性そのものであり、従って、クラス定義時（プログラム開発時）にも何等かの情報隠蔽をして直接にはデータ構造とメソッドを分けて書かなくても良くなるような仕掛が必要であることが分かろう。

2．　対象世界の一貫モデリング過程論

2．1　認識モデル（対象世界）の形成

そもそも人間は古来、その頭脳中で数え切れない程のモデリングの作業を行い、その結果を（広義な意味での）頭脳内再現シミュレーショ

ンを繰り返してきた。例えば、対象世界を見て、「これは何、あれは何」と解釈することもモデリングの一種である。もちろん必ずしもその過程全体を明示的（Explicit）に論理的に体系化してきたわけではなく、思考・思念を凝らし思いつくままに対象のイメージを頭脳内のどこかに暗示的（Implicit）に造り込み、変換し、誤解を正し、出来上がったイメージを自由に広義の意味でのシミュレーション駆動をさせてきた。これを明示的にモデル化したのが認知科学の分野における認識モデルである。本論文においては認識モデルとは人間の頭脳内に初めて生成された対象原世界のイメージのことを指すこととする。このモデルを「イメージ」とした理由は、明確ではあるけれどもまだ自然言語表現にさえなっていないという事情を表現するためである。我々の見解では、人間が認識できる（している）のは対象原世界（実世界）そのものではなく、人間が頭脳内に認識しているモデルこそ唯一の実世界である、と考えている。従って、この認識モデルを対象世界と呼び、モデリング過程の出発点の原点に位置する世界であると定義する。

2.2 素モデリングと概念モデリング段階

生成された対象世界（認識モデル）は、最終段階の計算機シミュレーションに向かって、モデル化過程を歩み始める。まず人間が最初のモデリングにおいてに必ずやることは認識モデルを自然言語（例えば日本語）に表現することであろう。これは別の言い方をすれば、広義の言語表現である人間のイメージ表現を自然言語に変換（言語変換）をしたものと見なすことも出来よう。この表現ではまず最初はいわゆる概念的なモデルを構成して表現することになろう。従って、このモデリング段階を概念モデリング段階と呼ぶことにする。

ここで、各モデリングにおいては図1のようなモデリングの要素が存在し、モデリングが行われていると考えている。即ち、前段階のモデル表現された世界をあるモデリング方法を使ってモデル化する。その結果、新たにモデル化された世界（対象世界モデル）が変換・構築される。モデリング方法及び、モデル世界を表現するためには広義の言語（表現系）を用いる。これがここのモデリングの際の構造である。

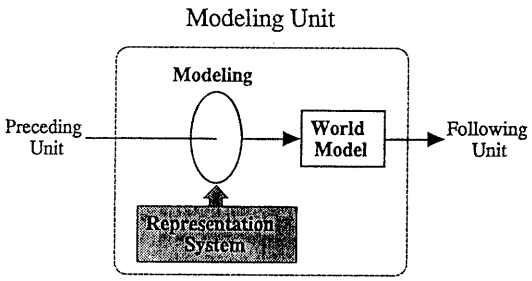

図1　モデル化単位の構造

2.3 論理モデリング段階

概念的なモデルで対象世界（認識モデル）を充分に写像し終わったならば、次に、その重要な本質的な部分を中心にして、より詳細にかつ論理的な、従って、それが解明されるべき専門分野の科学用語や理論や式（方程式など）を用いて表現される段階に来る。これを論理モデリング段階と呼ぼう。

この段階まででいわゆる分析モデリングと言われる段階は完了し、分析モデルは完成したと見なせる。この段階は計算機が出現する以前の科学・工学においても限りなく精密に行われて来ており、本研究で特に加えるべき知見等は何もない。ただ分析モデリング過程を明示的に記述したものである。この段階における対象世界に関する情報の記述は対象世界からの情報の落ちや追加、構造や機構の表現の歪み等は無しに必要十分に変換されて来たものと仮定する。

2.4 再構成モデリング段階

これ以降のモデリング過程は従って、十分に明示的に表現され、分析・分解され、如何様にでも再構成表現出来る状態にまで十分に理解された状態に置かれた対象世界を如何に正確に再構成し、計算機内で再現シミュレーションまで持ち込むかに掛かっている。この再構成過程はソフトウェア工学的な観点からみれば、以下の過程は設計過程とも言えよう。

対象世界の論理的なモデルが十分に分かっていることを前提とすると、再構成過程において

は対象世界の静的な構造部分と、動的な機構や振舞いの動きを表現する機構部分とに分けて再構成記述できるであろうと仮定して再構成モデルを構築する方針に仮定した。従って、その仮定に基づけば再構成モデリング段階は、再構成モデリング過程（段階）は再構成構造モデルと再構成機構モデルに分かれて記述されることになる。このモデリング段階では未だ原則として概念的・抽象的な計算機用語（情報用語）は用いられ、表現言語系としては実際のプログラミング言語が直接用いられず擬似的な言語表現に留まる。この段階はいわゆるシステム設計過程ということもできる段階でシステムの駆動に関するアーキテクチャの設計等も行われる。

2.5 再現モデリング段階

再構成（設計）過程が終わった対象世界は、設計仕様が決定され、その仕様にしたがい、計算機内への実現（＝実装）が、モデル論から言えば再現が、行われる段階に入る。この段階はソフトウェア工学的な詳細を言えばいくつかの段階に分かれるが、ここでは纏めて実現モデリング段階と考えておく。この段階のモデルでは前段階の再構成過程における構造モデルと機構モデルに分かれた記述を更に詳細化し、各々のモデルを実装（実現）モデルに変換し、最終過程の一つ前の過程でプログラムシステムとして記述して行くことになる。このステップで対象世界は静的な意味ではすべて表現され尽くされている筈である。対象世界のモデルのシステムとしての最終段階は駆動モデリング・ステップである。つまり、対象世界が再現されて起動され・駆動し始め、再現シミュレーションが始まるわけである。これを表現したものが対象世界に対して再現世界と呼ばれる。

2.6 一貫モデリング過程論

以上のモデリング過程を図式的表現法によって一つの図の形で表現し切ったものが図2の対象世界のモデリング過程である。図の上半分が分析モデリング過程、下半分が再構成（設計）過程、左上のアメーバ状の図形が対象原世界（実世界）、人間の絵がその内部にある頭脳内に表現された対象世界（認識モデル）を表現している。左下の図が再構成されて駆動している

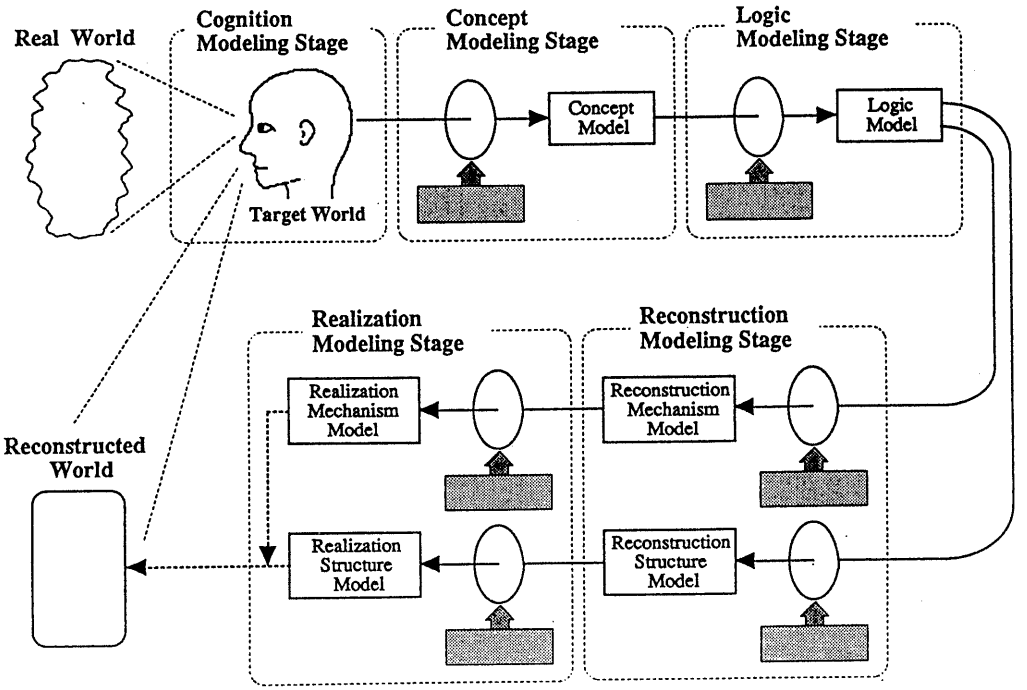

図2　対象世界のモデリング過程

対象世界の再現シミュレーションの図である。

このモデリング過程の図はモデリング方法（論）として仮定したものは何も無い。また如何なるプログラミング表現言語も仮定していない。人間が計算機を用いて対象世界を忠実にモデル化して言ったならばこのようなモデリング過程を通るであろうというもっともあり得るべき合理的なモデルか過程をつないでみたモデリング過程のフレームワークである、という意味を持つ。その意味ではきわめて一般的なまた汎用的な図であると共に、きわめて常識的な図でもある。この図は以下に展開される極めて柔軟性の高い、動的な変更の効くような高度な再現シミュレーションを実現する骨格を成すべき理論の核心部分を成す機構（オブジェクトベース機構）を自然にかつ合理的に導き出すための最も広義の枠組み・ガイドラインとして必須である。また実際に対象世界のモデリングを行い、再現シミュレーションを実現しようと試みる人達の道しるべとなる。実際に本論文の発展系の理論構築にも用いられつつある。

3. オブジェクトベース機構の理論と一つの実現

3.1 オブジェクト指向モデル ＝オブジェクトベース・モデル

本研究で用いたオブジェクトベースモデルは、相互関係を豊富かつ自然に表現できるオブジェクトベース－相互関係モデルを使う。構造の表現として（属性、データ構造）、構造を駆動させる機構としてメソッド、及び相互作用（通信機構を含む）から成るモデルを用いた。つまり、対外的な機能は相互関係の発現として捉えられる。

モデルオブジェクト ＝（属性、データ構造＋（メソッド）＋（相互関係）
（ここで相互関係
＝（相互関連リンク）＋（相互作用））

言い換えれば、オブジェクトベースとしてのオブジェクトは、モデル単位としての内部にデータ構造とメソッドを格納しており、メソッドはモデルオブジェクトの本来の構造に対する相互作用が生じたときには本来の構造が取るであろう反作用や振舞いを対内的・対外的に示せるように起動され、データ構造にアクセスして処理を行う。相互作用のメッセージ相互交換（通信）においては抽象化された”もの”対応のメッセージに組み立てられていなければならない。従って、メッセージ解釈系やメッセージ構築系が必要である。モデルとしてはこれで各オブジェクトが”もの”オブジェクトとして単独に駆動でき、多数のオブジェクトが相互作用（メッセージ通信）をしながら対象世界の振舞いを再現したシミュレーションになった、と考えられる。なお、シミュレーションにはC言語、およびC++言語が「実用上の観点」からみればベストであると考えられるのでそれを用いた。

3.2 オブジェクトベース機構

オブジェクトベース機構において持つべき構造や機能・機構的な側面の条件記すと、以下のようになる。

1. ＜データ構造とメソッド＞の分離構造を一体化（実体としてのカプセル化）させる静的な構造
2. 一体化オブジェクトに抽象的な名前をつけ、オブジェクト名もしくはそれとの関連名のみによってオブジェクトを選択し、それにメッセージを送ることで相互作用（アクセス）ができるようにする機構。
3. ユーザとはシステム的に隔離されていて、リモートアクセスのみが可能となるような機構を含む。
4. 一体化オブジェクト及びクラス構造に対するオブジェクト単位での持続的格納・検索・取り出し機構
5. 一体化した単位モデルオブジェクトの個々独立な振舞いを起動する動的な機構
6. モデルオブジェクトの抽象化表現機構。言い換えると、見え方やメンタルイメージとしてのカプセル化（情報隠蔽化）の機構。

上記の実現方法については多くの方法（言語、DB、OS等を用いて）が考えられる。以下ではシミュレーションに適した実現方法として考えだしたものを紹介する。

3.3 OODBMSを用いたオブジェクトベース機構の一つの実現

現状において最もオブジェクトベース機構を実現するのに利用し易いシステムの1つはOODBMSである。その理由は、システム及びアプリケーションの両領域で最も頻繁に一般に使用されている言語がC言語であり、C言語の構造体(struct)が使えることである。C言語のデータ構造体は前章までに述べた根本的な議論を除けば、本来の構造のデータ面での表現としては最も対象世界の構造を表現するだけの強い表現力と表現力の豊富さを持っていると考えられる。メソッドをうまく設定することで、本来的な構造の相互作用（前節で述べた全ての相互作用を含む、以下同様）の反応をかなり表現できる力を持っている。

しかし、OODBMSはあくまでデータを管理するシステムであり、メソッドを含んだオブジェクト単位での管理をするシステムではない。そこでOODBMSの機能を拡張することでオブジェクトベース機構を実現させることを考えた。本研究で基盤に用いたOODBMSはONTOS[5]である。ONTOSはC++言語を親言語とし、オブジェクトの構造をC++のクラス定義の形で記述し、メソッド（手続きプログラム）をC++のプログラムとして記述する。

まず、データ構造とメソッドを一体化して扱う静的な機構として、OODBMS内にメソッドのオブジェクトコードを格納することにし、ONTOSのアプリケーションプログラムからメソッドを含む持続的なクラス定義部分を切り離し、一体化されたオブジェクト格納場所であるOODBMS内に格納する機構を構築した。これにより、エンドユーザのアプリケーションプログラムからは常にプロセス間通信の形で外部からアクセスしなければならなくなり、オブジェクトの場所的な一体化が実現すると共に厳重なカプセル化が実現できた。外部からのアクセスにはOODBMSの標準機能であるSQL(ONTOS-SQL)があるので基本的にはこれを用いる。

一体化され、カプセル化され、持続的に（ディスクに）格納された各オブジェクトに対する起動（活性化）・メソッド起動・制御・操作に関しては、ONTOSにおいては指定オブジェクトに対してオブジェクト検索の副作用としてそのオブジェクトのメソッドを起動できる機能がある。これはONTOS-SQL[6]に装備された機能である。そこで、外部からのメッセージ通信（相互作用）によりオブジェクトのメソッドを呼び出せるようにすることにより、OODBMS内のオブジェクトをオブジェクトベース的な意味でのオブジェクトとして取り扱えるようにすることを考え出した。そのためONTOS-SQLを拡張したE-OSQL（拡張ONTOS-SQL）を製作した。

また、SQLは外部からアクセス出来るようにプロセス間通信によって切り離されている。ところで、メッセージ通信の形式は単純に文字列のやり取りのみであるのでエンドユーザプログラムをクライアントとし、オブジェクト側のE-OSQLプログラムをサーバプログラムとする、クライアントーサーバ方式を採用したシステムとして構築した。このようにしてユーザとオブジェクトが双方向にやり取りできるシステムが構築できた。また、クライアントプログラムは文字列をやり取りできるシステムなら何でも良い。従って、シェルやFortran等の任意のプログラムからアクセス出来る。

図3にE-OSQLによるオブジェクトベース機構の内部構成を示してある。また、E-OSQLの

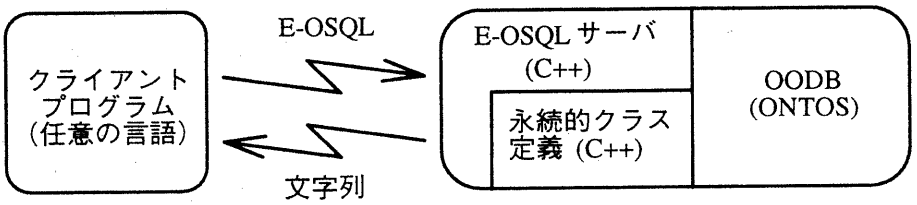

図3　オブジェクトベース機構の構成

```
SQL> insert into Macro_mean values();
#1Macro_mean;
SQL> select tunnel().init() from Macro_mean where localName() = '#1Macro_mean";
SQL> select calc(100,100) from Macro_mean where localName() = "#1Macro_mean";
SQL> select output() from Macro_mean where localName() = "#1Macro_mean";
field
200 100 1
191 913 3 -4 311
        .
        .
```

図4　E-OSQLの例文

例を図4に示してある。図4における構文は”もの”オブジェクトに対するものとしてはメンタルリアリティを損なうので高度なユーザインターフェイスの仕掛が必要である[7]。

以上により、OODBMSを基盤としたオブジェクトベース機構はオブジェクトベース機構の成立条件を満たしており、実際に一つの実現方法としてOODBMSを用いたオブジェクトベース機構が考案され、利用が可能になった。

4．オブジェクトベース機構を用いたシミュレーション

4.1　シミュレーション機構としてのオブジェクトベース機構の使い方

オブジェクトベース機構とは簡単に言えば、対象世界の”物”をモデル化し、”もの”単位での駆動を可能にし、多数のオブジェクト群の間での相互作用を各オブジェクト自身が処理しながら、オブジェクトベース世界全体が整合的に振舞って行く、それを実現する機構である。従って、オブジェクトベース世界を素直に構築して行くだけで良く、手続き的にどの様に動かすか等ということは考えなくて良い。つまり、メインプログラムを書かない。また、モデル上及び駆動操作上はオブジェクトベース機構を意識することは全く無くて良い。オブジェクトの定義時にはONTOSやE-OSQLについて考慮してプログラミングする必要がある。（この辺りはまだ改良の余地が多い。直接構築する試みも始まっている[8]。）　オブジェクト（クラス）定義をして行き、オブジェクト間の相互作用リンクを張って行くことでオブジェクトベース世界は出来上がる。

4.2　オブジェクトベース希薄気体数値風胴の概要と駆動

オブジェクトベース機構の有効性を実証するために我々は以前から開発してきていた希薄気体数値風胴シミュレーションプログラムをオブジェクトベース機構に搭載出来るように改造し、多くの駆動例を作ってその効果等を調べた。以下にまず希薄気体数値風胴のごく簡単な概要を示す。

希薄気体とは空気の分子単位での流れと見なしても良いほどの希薄な気体の流れであり、分子単位で扱う。風胴とは通常矩形断面の直線風路を作りその中に飛行物体等を置いて風を起こし（流し）、飛行物体等が作る流れの様子を観察・計測する装置である。数値風胴とはこれらの物理実験をすべてシミュレーションモデルで置き換えて計算機内で再現シミュレーションしようというものである。希薄気体の流れの数値シミュレーションには殆どDSMC法が用いられる。このDSMC法においては、風胴壁（上下流の境界も含む）、分子、飛行体がオブジェクトであり、オブジェクト間に、衝突をその相互作用のメソッド（手続き）の内容とする相互作用リンクを張ることでモデル（実現モデル）は成立する。DSMC法は元々直接シミュレーション法であるので、オブジェクト指向の観点からすると、オブジェクトとその相互関係（相互関連＋相互作用）の設定を行えばそれで完了である。図5にOMT法で記述された数値風胴のオブジェクトモデルダイアグラムを示した。前節に述べたようにこのダイアグラムを詳細化して行くことでオブジェクトベース数値風胴の実現に至る。

オブジェクトベース機構を用いた数値風胴のシミュレーション結果は示さない。なぜならば、

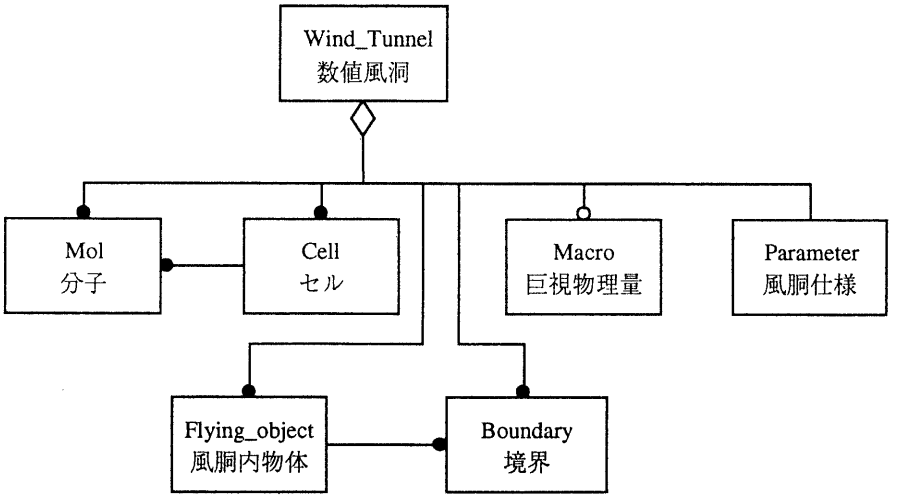

図5　数値風胴に対するOMT表現のオブジェクトダイアグラム

数値シミュレーション結果そのものは従来と変わるはずが無いからであり、実際に実験、他の方法による計算結果と何れも良く合った結果を示し、計算方式が正当なことが裏付けられた。図6にオブジェクトベースのシミュレーション中の計算機のCRT画面の例の写真を示す。次節で述べるオブジェクトベースシミュレーションの特性はこの数値風胴の運用の結果得られたものである。

4.3　オブジェクトベースシミュレーションの特徴

現在までにオブジェクトベース数値風胴の駆動例の結果から次の結論が得られた。

1．複雑な対象世界になればなるほど手続き型シミュレーションより有利になる。プログラミングや操作の手間が手続き型のプログラムほどには増えない。「抽象化による複雑さの管理のしやすさ」の結果が現れたとも言えよう。
2．対象世界の構造や駆動の臨機な変更が任意かつ即座・動的に実現・再駆動開始可能になった。
3．オブジェクトが「モデルの枠内であるが」”物と同等なもの”して取り扱える様になった。
4．自然なカプセル化（情報隠蔽）とオブジェクト間インターフェイス機構が実現。

5．　オブジェクトベース機構の評価と議論

5.1　静的・動的な組み合せの多様性及び制御・操作の柔軟性が大きい理由

これらは本質的には、オブジェクトベースアーキテクチャが手続きアーキテクチャよりも元になっている再構成モデルにより近い、つまり、構造的な記述を行うアーキテクチャに近づいたためであろうと考えている。つまり、構造を表現するためにその構造物が対外的に行うであろう反応（相互作用）をメソッドの形で記述してそれを起動し、構造物が外的な相互作用に対する振舞いをさせることで「構造物の代替」機能をもたせているに過ぎない。本来、構造物が外的な刺激（相互作用）に対して持つ振舞いや特性はいわば無限にあり、それを小数の振舞いだけを持つとした<データ＋メソッド>の一体化の仕掛は構造物の振舞いの多様性を極めて小数の場合に制限したことになり、モデル化の常とはいえ本来モデル化で省略したくない場合までも切り捨ててしまっている。本オブジェクトベース機構においては、擬似的に構造物の部分をカプセル化し、オブジェクトとして（構造物により近い形で）扱える様にしたことで上記のような多様性や動的な変更の多様性の特徴を、手続き型システムに比較すればかなり回復したものであるといえよう。

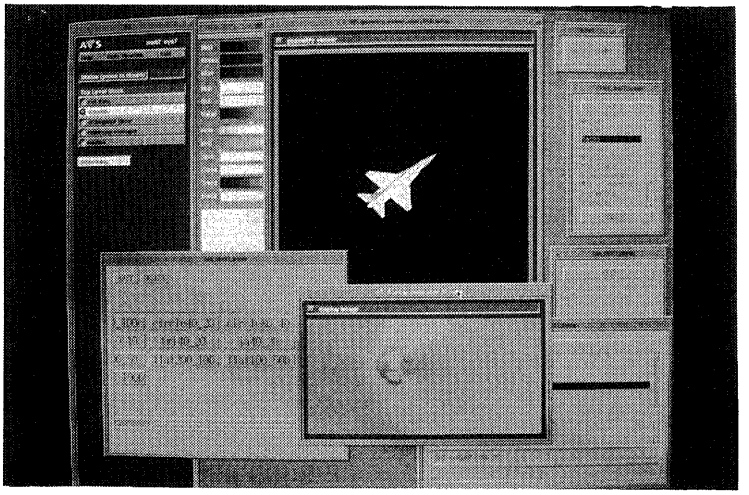

図6　オブジェクトベースシミュレーションの駆動場面

オブジェクトベースシミュレーションにおいては、実現されたプログラムの駆動という側面からみると、オブジェクト間の相互作用の仕掛そのものだけがプログラムされていて、その相互作用の発現・発揮・起動シーケンス（順序及び分岐）は特別の場合を除いて一切記述されていない。また、これを支援するかのように、動的結合機能がオブジェクト指向言語には標準的に提供されている。つまり、オブジェクトというモデル単体の振舞いとオブジェクト間の相互作用ネットワークのみが決められていて、そのネットワーク（ここでのネットワークとは振舞い起動の順序可能を示すのみ）の起動・駆動順序は一切決められていない。従って、各オブジェクトに制御権が行ったときに、まさにその時点で選択枝のどれにするかを決めることが出来るという特徴（動的選択）つまり、駆動の変更の柔軟性が常に充分確保されていることになる。

但し、効率的な面では言うと、手続き的な追跡を行うと分かることであるが、しばしば、効率的には問題となる処理を実行している。従って、オブジェクトベース機構を用いた処理効率は「同じ実行を行う」場合を比較すると、少なくとも数十％以上効率に関しては手続き型が優る。オブジェクトベースアーキテクチャは効率よりも、実質的機構上及び見え方の点において”もの”としての振舞いを実現し、かつ”もの”が本来もつ振舞いや諸特性の多様性や動的な変更のしやすさと「らしさ」を狙ったものであることがわかる。

5．2　オブジェクトベース機構のその他のメリット

オブジェクトベース機構はモデル化されたオブジェクトベース世界のクラスやインスタンスを多数持ち、これらを持続的に合理的かつ効率的に管理する、という側面においてもそのメリットを発揮する。即ち、

1．ドメインユーザにとってみれば自己のメンタルイメージにある対象世界に従来よりもはるかに近いイメージでモデル化されたクラスやインスタンス（数値シミュレーション結果例）を保存できることになる点が大きなメリットであろう。そのイメージは”もの”に近いメンタルリアリティを持てることになろう。また、システム開発者やシステム屋も人間であるから、やはり物理的世界のものに対応するモジュールがシステム内にある、と考えてシステムのメンテナンスや改訂版の作成をすることは神経的・精神的に楽なのではないであろうか。そのこと

は、より高度なシステム構築につながることを意味するのではないであろうか？

2．システム的な面からみてもオブジェクトベース世界の多くのクラスやインスタンスの管理に関して従来型の何等かの管理機構に加えて、意味関連を持つような抽象的なある種の意味ネットワークのノードとしての意味を持って張られるわけであるから、新たにもう一つの管理機構が見つかったのと同じ様な効果を持つ。

6．現状での結論と今後の研究開発の展望

実装した現状の機構の評価としては、

- オブジェクトベースとしての機能は十分発揮している。オブジェクトベースモデルを更に精密にすることでより高度な発展も望めるので改良中である。
- ユーザインターフェイスの部分がまだまだ十分ではない。これに関しては相互作用支援環境として現在開発中である。これが出来上がれば、いわゆるメンタルリアリティもはるかに良くなり、オブジェクトベース世界の実在が信じられるようになるかも知れない。本質的ではないし、我々の研究方向とは異なっているが、VR の技術を取り入れる余地も大きい。
- 起動・駆動・操作中はともかく、オブジェクトの生成過程がまだソースコードの直接プログラミングのみに頼っているので、これを例えば、直接”もの”オブジェクトを生成できるように改良する必要がある。オブジェクトの直接生成が望ましい。OBCADS が一部実現しつつあるが [8].

今後の展望としては

- 実装方法は種々の形式が考えられるので、よりオブジェクトベース的な実装、より大きい多様性や柔軟性を持つ方法を議論、研究開発しているところである。
- オブジェクトベースモデル、及び計算モデルとしては現状で良いのか否かの再検討も必要である。現在検討中である。

参考文献

[1] 所真理雄他編、「オブジェクト指向コンピューティング（岩波コンピュータサイエンスシリーズ）」、岩波書店、1993年 11月。

[2] J．ランボー他著、羽生田栄一監訳、「オブジェクト指向方法論--モデル化と設計」、トッパン、1992年 7月。

[3] 米沢明憲、柴山悦哉、「モデルと表現（岩波講座ソフトウェア科学１７）」、岩波書店、1992年 4月。

[4] 畠山正行、金子　勇、上原　均、「擬似オブジェクトベース機構の基づく数値シミュレーション」、第 12 回シミュレーションテクノロジーコンファレンス、シミュレーション学会、pp.317-320、1993年 6月。

[5] 宇田川佳久、「オブジェクト指向データベース入門」、㈱ソフトリサーチセンター、1992年 8月。

[6] ONTOS SQL ガイド（ONTOS DB 2.1 日本語マニュアル）, 日商エレクトロニクス（株）, 1991年。

[7] 畠山正行、上原　均、「オブジェクトベース・シミュレーションの GUI 実行支援環境」、情報処理学会第 51 回ハイパフォーマンスコンピューティング研究会、平成 6年 6月 17日。

[8] 畠山正行、小林秀行、「オブジェクトベース CADS システムの構築、-- OODBMS を基礎にした設計・加工及びシミュレーションシステム」、第48回情報処理学会全国大会講演論文集、pp.6-181～6-182、1994年 3月。