プログラミング 1−5
（1995. 6. 16）

オブジェクトベースモデルに基づく シミュレーション駆動機構の構築

金子 勇　畠山正行
茨城大学工学部情報工学科

本研究の目標は、対象世界のモデル化から実装・駆動までを全てオブジェクト単位で行うというオブジェクトベースシミュレーションを行う際の、オブジェクトの駆動機構として、オブジェクトベース機構を設計・構築することである。そのために、オブジェクトベースシミュレーションを行う際のモデル化の基本となるオブジェクトベースモデルを新たに規定し、それを基礎としてプロトタイプベース・永続的・並行・ストリームベースという特徴を持つオブジェクトベース機構モデルを作成し、オブジェクトベース機構を実際に実装した。このオブジェクトベース機構は、UNIX システムに対してオブジェクトベース機構モデルを対応させた機構であり、UNIX システムをオブジェクトシステムとして利用することが可能となる。このオブジェクトベース機構を用いたシミュレーションシステム例も完成しており、有用性が確認できている。

A Simulation Drive Mechanism Based on the Object-Based Model

Isamu KANEKO　Masayuki HATAKEYAMA
Ibaraki University

In the present study, we have newly developed the Object-Based mechanism (OB mechanism) for the high quality simulations in the fields of the natural phenomena or the engineering structures. In the past study, we have already developed an Object-Based mechanism based on the Object Oriented DataBase Management System (old OB mechanism). The old OB mechanism has, however, some defects; for example, since the transaction processings concentrate on the object management server on the OODBMS, the load of the CPU is comparatively heavy. Since the old OB mechanism has the class based, sequential object model, the more reasonable computing models for our simulations must be considered. Therefore, a new Object-Based model and a new Object-Based mechanism model are designed, developed and implemented. This newly implemented OB mechanism is applied to constitute the new simulation support environments, and this new simulation system (computing environments) shows various validity to simulate and analysis the target natural phenomena.

1 はじめに

我々の研究グループでは流体力学（希薄気体力学）分野といった、工学での分析・解析対象となるような現象世界のシミュレーションシステム及びその支援環境について研究している。このような解析シミュレーションにおける対象世界のモデル化手法には、オブジェクト指向モデリングパラダイムによるモデル化が有効である[1]。これは、データと振舞い（操作）が一体となったオブジェクトをモデル化の単位として、対象世界をコンピュータ内部に再現をすることによってコンピュータ内でも現実世界と同様に物体を扱うことができるというものである。我々は、このようなモデリングパラダイムに基づくシミュレーションを「オブジェクトベースシミュレーション」と呼んでいる[1,2]。このオブジェクトベースシミュレーションでは、オブジェクトをモデル化の単位とし、実現及びその駆動という対象世界の再現の末端まで一貫してモデルの精度を保つというような自然なモデリングを行うことができる。しかし、一般の応用分野では、この手法を用いることは殆ど実現されていない。

この「オブジェクトベース一貫モデリング」が行われない原因として次のことが考えられる。モデリングの方法は手続き指向プログラミングを用いることが多く、このモデリングの実装過程（再現モデリング段階）[3]においてモデルの属性・振舞い・相互作用といったものが上流のオブジェクトベースとは異なるパラダイムに基づくモデルによって記述されてしまう。その結果、モデリング過程において有効でかつ自然な概念であるオブジェクト指向が応用分野では殆どの場合、充分に効果的な活用がなされていないというのが現状である。

そこで本研究では、オブジェクトベースシミュレーションを行うためのより良い環境を構築するために、オブジェクトベースモデル[3,4]を基礎とするオブジェクトベース機構モデルを構成し、実際に UNIX システム上に実装することで、従来環境との整合性に優れた、より実用的なオブジェクトベース機構を実装する計画をたて、それを実現した。以下はその報告である。

2 OODB を用いたオブジェクトベース機構

以前の研究で我々は、オブジェクトベース機構（以下 OB 機構）を考案した[1]。OB 機構の目的は、オブジェクト指向モデリングを用いてモデリングする際のモデル化の単位であるオブジェクトをコンピュータ上で管理・起動することである。

この OB 機構を、参考文献[1]ではオブジェクト指向データベースシステム（OODB）を用いることで構築した（OODB-OB 機構）。C++のようなコンパイラ言語をベースとする、一般的な OODB ではオブジェクトの静的データ構造は管理するものの、オブジェクトの動的なメソッドを管理することができない。OODB をシミュレーション環境として利用するために、OODB にオブジェクトの振舞いを管理するメソッド管理サーバを追加し、このサーバへのプロトコルとして EOSQL を作成することで、オブジェクトの振舞いの管理も行なえるようにしたのが OODB-OB 機構であった。

この OODB-OB 機構を利用することで、対象世界の物体をモデル化したオブジェクトがその振舞いも含めて管理可能となり、従来よりもシミュレーション対象世界の構築をはるかに容易とすることが可能となった[1,2]。

しかし、このシステムにも速度面やオブジェクト及びシステムの構成面などでの欠点が目立つようになり、また、よりオブジェクトベースシミュレーション向きの機構及び駆動環境が求められるようになってきた。

OODB-OB 機構の問題点

- クライアント・サーバ方式により、オブジェクトのデータや振舞いを統一的に集

中管理していたため、クラス構成に変化が起きる度にそのつどデータベーススキーマの変更が必要となることや、オブジェクトの振舞いがメソッド管理サーバに集中していることによる負荷の増大など。

- OODB-OB 機構では C++をベース言語とする OODB（ONTOS）を用いているため、オブジェクトモデルとして C++のオブジェクトモデル（逐次モデル、クラスベースモデル）を採用しているが、他のオブジェクトモデルを使用したい。
- OODB を基盤としているため、従来のシミュレーション環境との整合性に問題がある。

3 オブジェクトベース機構モデル

我々の研究グループでは、対象世界における“もの”と常に一対一対応させたモデルを作り、それを同一（同等）内容の別表現に変換するモデリングを行って対象世界を写像し、この変換（写像）を必要な回数繰り返すことで、最終的には元の世界の同等内容の別表現をコンピュータ上に再現するという「オブジェクトベース一貫モデリング過程論」を提唱している[3]。

オブジェクトベース機構は、このモデリング過程の最後の、再現及び駆動モデリング段階を実現するための機構である。そのような実現のためには、OB 機構のオブジェクトモデルとしては、再現及び駆動モデリング段階の直前の再構成モデリング段階におけるオブジェクトベースモデルをより具体化させたオブジェクトモデルが必要である。そこで、以下ではまず再構成モデリング段階のオブジェクトベースモデルを記述し、それを基にしたオブジェクトベース機構モデルを構築することとする。

3.1 オブジェクトベースモデル

再構成モデリング段階のオブジェクトベースモデルは以下のように定義されている[4]。

OB モデルオブジェクト
＝（属性、データ構造）
＋（振舞い記述）＋（相互関係）

ここで

相互関係
＝（相互関連リンク（静的））
＋（相互作用（動的））
＋（OB インターフェイス）

オブジェクトベースモデルでは、一旦は周囲から切り離されたモデル化最小単位を用いるため、オブジェクト間の相互関係の定義と設定は他のオブジェクト指向モデルに比べてもより重要である。相互関係は三種類の要素からなる。

相互関連リンク

オブジェクト間の静的な相互関係を表現する。この相互関連リンクは、例えば、相互作用の発現の際の相互作用（力学）計算をする際に活性化され（本来はその時点で動的に）参照され、相互作用の影響が計算される。

相互作用

メソッドを発動・起動させ、自オブジェクトの振舞いを発現させ、それを他オブジェクトに伝える仕掛である。本来は、メソッド発動のタイミング・発動条件・発動手順・発動構造（発動各要素間の静的関連）・発現（起動）シーケンス等を記述することでオブジェクトの動作や振舞いを生じさせ、他オブジェクトへ伝達し、その影響を相互に及ぼし合うことになる。しかし、実際には上記のオブジェクトの動的な事柄に関係する構造部分とのかかわりは省略され、結果としての動作や振舞いのシーケンスを結果論的に記述し、結果としての変化量を計算する表現になったメソッドが多い。

OB インターフェイス

他のオブジェクトとのコミュニケーションのプロトコルの設計。オブジェクト間では

相互作用計算のためのデータを相手に要求し、その計算結果のうち相手オブジェクトに関係するデータを送り返すというのが基本プロトコルである。

3.2 オブジェクトベース機構モデル

オブジェクトベースモデルを基に、オブジェクトベース機構のモデルであるオブジェクトベース機構モデルを構築する。本モデルは再現モデリング段階の実装（再現）モデルの最初のステップに相当する。

オブジェクトベース機構モデルでは

オブジェクト
＝（データ）＋（メソッド）
＋（プロセッサ）
＋（オブジェクトリンク）

オブジェクトリンク
＝（静的リンク（集約 or 関連））
＋（動的リンク）
＋（メッセージ解釈機構）

である。

シミュレーション環境としての要求を考慮し、本オブジェクト機構モデルの実装上の特性モデルとして、プロトタイプベース、永続的、並行（マルチスレッド）、ストリームに基づくメッセージのオブジェクトモデルを採用する。

プロトタイプベースのオブジェクトは、Self[5]などに見られるものであり、クラスという概念が存在しない。振舞いの共有は、委譲によりインスタンスレベルで行われ、オブジェクトの生成はプロトタイプとするインスタンスをコピーすることで生成される。本研究で対象としている、試行錯誤を基本とするシミュレーション環境では、型とするオブジェクトをコピーし、多少の変更を加えた後に直ちに駆動させることでシミュレーションを行うことが多いため、このオブジェクトモデルが有効となる。

次に、オブジェクトの永続性は OB 機構の目的が、シミュレーション時のオブジェクトを、管理・駆動させることであったため、必須の条件となる。

また、シミュレーションの観点からオブジェクトの並行性は自然現象の本質の自然なモデリングであり、特に望まれるものである。本モデルでは、オブジェクト内部のメソッドも並行性をもつマルチスレッドモデルとした。

他に、オブジェクトベースモデルでは、オブジェクト間の相互作用を、パケットタイプによるメッセージによるものではなく、他のオブジェクト間の関係と同様に、リンクの一種としている。そのため、本モデルではオブジェクト間にメソッド起動などの相互作用が起こる際には、オブジェクト間に通信のためのストリームを生成するという、ストリームに基づくメッセージモデルを採用した。

3.2.1 オブジェクトの構造モデル

これらをふまえた上で、次のようにオブジェクト構造モデルを設計した（図1）。

- オブジェクトは、一意のオブジェクト識別子(oid)とポートを持つ
- 集約オブジェクトは任意個数のスロットを持つ
- スロットは、スロット名とオブジェクトポインタを持つ
- スロットには関連スロットと集約スロットがある
- データオブジェクトは任意長のビットデータを持つ
- メソッドオブジェクトは任意長のビットデータであり、プロセスオブジェクトによって実行可能なデータ列である
- プロセスオブジェクトは内部に仮想的なプロセッサを持つ

このモデルでは、オブジェクトは本質的に、集約スロットによる階層構造を有することとなる。

3.2.2 オブジェクトの振舞いモデル

次に、前節の構造モデルを駆動するオブ

ジェクトの振舞いモデルを記述する。

- プロセスオブジェクトはメソッドオブジェクトを解釈することによってデータオブジェクトやポートに対してアクセスを行う
- オブジェクトのポートは、実際にはオブジェクト内部に集約されている active プロセスオブジェクト(メッセージ解釈プロセス) のポートである。active プロセスオブジェクトのポートは自分自身で持っている
- オブジェクト相互作用の手順 (図 2)
  1. 活性化オブジェクトに接続を依頼する (①)
  2. 活性化オブジェクトは、依頼先のオブジェクトが活性化 (active プロセスが集約) していなければ、active プロセスを生成する (これによりオブジェクトのポートが生成される) (②)
  3. 活性化オブジェクトは依頼元と active プロセスのポートを接続する (③)
  4. プロセスオブジェクト間のやりとりによって、メソッドが実行される (④)
  5. active プロセスオブジェクトによって、他のプロセスオブジェクトが生成されることもある (⑤)
  6. 相互作用のリンクが全て切断された時点で、active プロセスが終了し、オブジェクトが非活性化される

このモデルでは、集約オブジェクトは一種のチューリングマシンと見なすことができる。

3.2.3 オブジェクト生成と委譲

オブジェクトは、型とするオブジェクトの全スロットをコピーすることで生成される。集約スロットの場合、ポインタが指し示している、集約の子オブジェクトも再帰的にコピーする。

実際には、オブジェクト生成時はコピーを行わないで、スロットポインタ先を書き換える際に初めてコピーが行われる (書き込み時コピー)。書換え前は、コピー元のスロットポインタ先が読み込まれる。これにより、複数のオブジェクトでスロットが共有でき

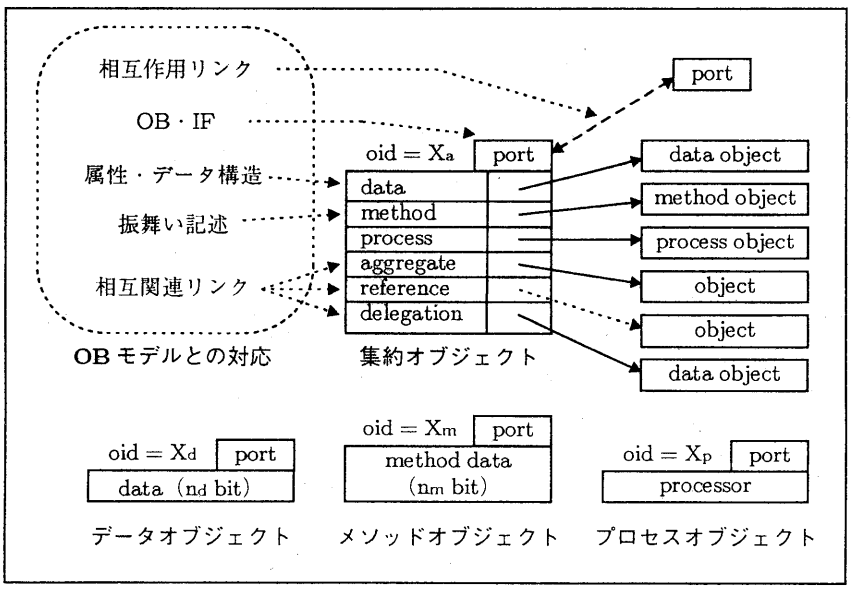

図 1 オブジェクト構造モデル

ることとなる。また、これによりメソッドの委譲も実現されることになる。

3.2.4 オブジェクトの名前付け

集約スロットによるポインタは必ず木構造を取ることとする。そのため、全てのオブジェクトの名前は、rootオブジェクトからの、集約スロットのスロット名の合成で、一意に決定することができる。また、関連スロットを用いる名前も使用可能である。

4 オブジェクトベース機構の実装

オブジェクトベース機構モデルに基づいた新しいOB機構を実際に実装した。実装はUNIX OS上で行い、HP9000(HP-UX9.05)上やLUNA-88K(UNIOS Mach2.5)などのプラットフォーム上で稼働している。

4.1 実装方針

2章で述べたように、OODB-OB機構は、OODBをオブジェクトの保存場所としているため、試行錯誤を基本とするシミュレーション環境として問題があった。

そこで、本研究では実用的なシミュレーション環境を目指し、UNIXのファイル・プロセス機構を極力用いるという方針で実装を行った。

4.1.1 オブジェクトの実装方針

- UNIXのディレクトリを集約オブジェクトとして用いる。
- 通常ファイルをデータオブジェクト、実行ファイルをメソッドオブジェクト、UNIXプロセスをプロセスオブジェクトとして用いる。
- 粒度の細かいオブジェクトのために、activeプロセス内のアドレス空間内にもオブジェクトを生成できるようにし、それらを一つのファイルに出力することで永続性を持たせる。
- portとしては、基本的にUNIXドメインのTCPバークレイソケットを用いる。
- UNIXファイルを全てオブジェクト扱いとする。
- オブジェクトの生成はプロトタイプの

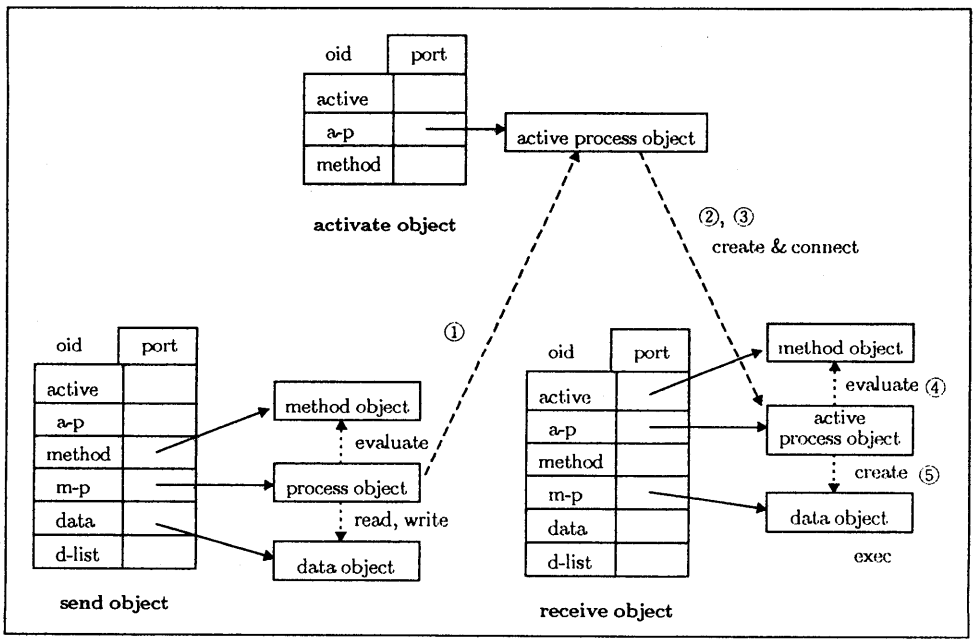

図2 オブジェクト間の相互作用

copy メソッドを起動することで行われる（通常、委譲機構で処理が行われる）。実際には、書き込み時コピーなので、copy メソッドは委譲スロットの変更などのみ行う。

4.1.2 オブジェクト振舞い機構の実装方針

- オブジェクトの振舞いは、メソッドオブジェクト（実体は実行ファイル）を実行することで実現される。
- 実際にメソッドオブジェクトを実行させるのは、同一オブジェクト内の active プロセスである。
- オブジェクトの活性化(active プロセスの生成）は活性化オブジェクト内の active プロセスが行う。このプロセスは常に動作している。
- active プロセスは委譲スロット(実体は特定の名前のファイル）を参照し、その中から実行可能なファイルを検索することで委譲機能が実現される。すなわち、UNIX における実行検索 PATH がディレクトリごとに持っていることになる。この機構は通常ファイルにも働く。
- メソッドの起動は基本的に実行ファイルの実行であるが、active プロセス内の関数の実行もメソッドの起動と見なすこととする。これにより、プログラムの実行と関数の実行が、共にオブジェクトのメソッドの起動として扱うことができる。

4.2 実装

基本的な実装は、C 言語のライブラリの形で行われている。ライブラリ内の基本的な機能は、他（もしくは自身）のオブジェクトのメソッド起動であり、

```
FILE *fp = OB_Fcall("r", OB_self, method);
```

などの形で用いる。これにより、オブジェクトのメソッド起動(プログラムもしくは関数の実行）やファイルなどの UNIX リソースへのアクセスが全てオブジェクトのメソッド実行として行える。ライブラリには、実際にはメソッドの引数や戻り値の指示、もしくはメソッド起動のタイプ（過去、現在、ストリーム呼び出しなど）により複数存在する。

メソッドの作成は基本的に実行プログラムの作成、もしくは active メソッドにリンクする関数の生成であり、従来の開発環境を用いて行うこととなる。

ライブラリ以外に、基本的なテンプレートオブジェクトが作成されており、オブジェクトの生成や削除などのメソッドはこれらのオブジェクトが保持している。これらのオブジェクトのコピーで新しいオブジェクトを生成することで、各スロットが共有され、またそれらのオブジェクトを基に新しいオブジェクトの生成が容易に可能となっている。

4.3 オブジェクトの駆動

ユーザがオブジェクトを利用するための専用のシェルが作成されており、

```
% - object/method arg1 arg2 ...
```

などとして用いる CUI や、Macintosh のフォルダに見られるようなアイコン型の GUI、もしくは、全オブジェクトに三次元形状を持たせることで、オブジェクト空間を作成し、そこでのエージェントの形でのユーザインターフェイスなどが作成されている[6]。

これによりユーザは、プログラムの実行やオブジェクトの生成などを、オブジェクトのメソッド起動として扱う。

5 実用性の検証

このオブジェクトベース機構を用いたシミュレーションシステムとして、流体計算を直接シミュレーションモンテカルロ法により行う一貫駆動支援環境などがすでに完成しており、物体定義からシミュレーション結果の可視化まで一貫して行える柔軟なシステムを構築できることが確認されている[7]。この一貫駆動支援環境における再構成モデリング段階の対象世界全体のオブジェクトモデル構成を図 3 に挙げる。

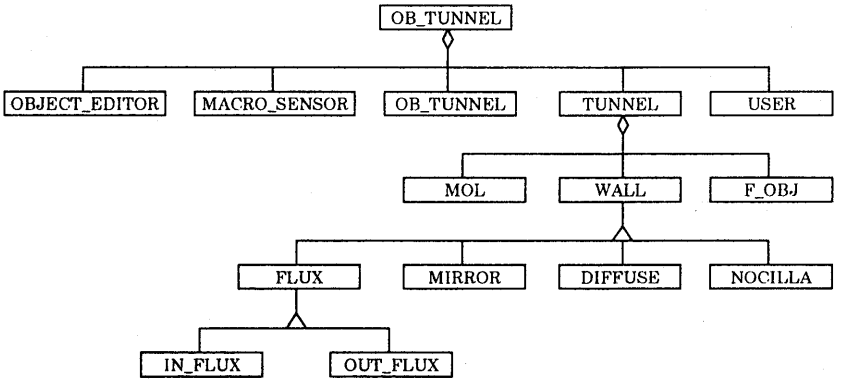

図3　一貫駆動支援環境における再構成オブジェクトモデル（OMT 表記）

6　結論

本研究では、以下の新たな成果が得られた。

1. オブジェクトベースモデルに基づく、より合理的でシミュレーション向きのオブジェクトベース機構モデルの提案と構築を行い、そのモデル適性を確認した。
2. オブジェクトベース機構モデルに基づく、オブジェクトベース機構の実装を行い、その有用性を確認した。
3. オブジェクトベース機構モデルの適用により、UNIX システムをオブジェクトシステムとして利用可能となった。
4. 数値風胴への適用によるオブジェクトベース機構の実用性及び有用性を立証した。

本研究の成果により「オブジェクトベース」モデリング概念構想にその源を発する我々の研究は、OB モデル、OB 一貫モデリング過程、OB 機構、OB シミュレーション、OB 駆動環境などを実現することにより、その体系的なシミュレーションアーキテクチャを一通り示したことになると確信する。

参考文献

[1] 畠山正行、金子 勇、「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」、情報処理学会第 17 回プログラミング研究会研究報告、Vol. 94, No. 49, pp.33-44, 1994 年 6 月 3 日。

[2] 畠山正行、金子 勇、「オブジェクトベース機構に基づく数値シミュレーション」、情報処理学会第 51 回プログラミング研究会研究報告、Vol. 94, No. 51, pp.1-8, 1994 年 6 月 17 日。

[3] 畠山正行、「高度なシミュレーションのためのオブジェクトベース一貫モデリング過程論とその駆動支援環境」、第 1 回情報処理学会数理モデルと問題解決研究会報告、Vol.95, No.44, pp.33-40、1995 年 5 月 18 日。

[4] 畠山正行、「流れの方程式のオブジェクトベース一貫モデリングと直接シミュレーションの実現」、第 56 回情報処理学会ハイパフォーマンスコンピューティング研究会、1995 年 6 月 1 日。

[5] David Ungar, Randall B.Smith, "Self : The Power of Simplicity", OOPSLA 1987.

[6] 上原 均、畠山正行、「Object Reality Interface 機構モデルの概念設計と実装例」、情報処理学会第 57 回ヒューマンインタフェース研究会報告、Vol.94, No.57, pp.17～24、1994 年 11 月 10 日。

[7] M.Hatakeyama, K.Akita, Y.Hasegawa, N.Takimoto, M.Watanabe, "Object-Based Numerical Wind Tunnel System with Integrated Support Environments", Proceedings of the ICES '95, Hawaii, USA, Jul. 30 - Aug. 3, 1995 (in printing).