データベースシステム 103－1
（１９９５．５．２５）

オブジェクト管理システムとしての オブジェクトベースリポジトリの考察

小林秀行　畠山正行
茨城大学工学部情報工学科

オブジェクトベース・リポジトリ（OBR）とは、<<データ構造>+<メソッド>>で定義されるオブジェクトを「モデル及び実現の単位」として、オブジェクト生成から全てのハンドリングの過程において常時必ず扱わなければならないというシステム内において、常時オブジェクト単位での管理や扱いが出来る様に多様なリポジション機能を持った支援環境の中核的システムとして考案されたものである。OBRは完成品のオブジェクトだけではなく、オブジェクトの内部要素、未完成オブジェクト、データ、等のおよそオブジェクトに関するあらゆる「静的な関係ぶつ」を格納・管理・検索・取り出し、そしてオブジェクト単位での起動が出来ることを目標としたシステムである。

本報告においてはオブジェクト管理システムとしてのOBRを、OODBやソフトウェアリポジトリ、及びOB機構等を比較対象システムとし、その構成や機能の違い・共通点、将来的な発展の展望等を考察・議論した。

Object-Based Repository as Object Management System

Hideyuki Kobayashi　Masayuki Hatakeyama
Ibaraki University

In the development processes of the software modules based on the object-oriented paradigm, a huge variety of objects, parts of objects, incomplete/complete objects, complete/incomplete data, the connections among objects, and their huge variety of fragments/combinations are generated and made use of. To manage these various objects and other object modules and fragments, we have developed a repository system for these various objects. We call this support environment the Object-Based Repository (OBR).

In the present paper, we will try to compare this OBR system with other reposition systems or reposition support environments such as the OODBs, the software repository system, and the Object-Based mechanism. We also discuss the features, the characteristics and differences among those systems. We conclude that the OBR system is most important and hopeful than other reposition systems as the fundamental system for the Object-Based simulation and its support environment.

1 はじめに

我々の研究グループでは、以前からオブジェクト指向（正確にはオブジェクトベース）のシミュレーションシステム及びその支援環境を開発・研究している。その一環として「オブジェクトベースシミュレーションに用いるためのオブジェクト」を基本とするオブジェクトの生成支援環境を構築中である [1, 2]。しかし、この生成支援環境の研究・開発の進展にともない、リポジション機能に関する多様かつ複雑・煩雑な要求が生じて来るようになった。例えば、生成された完成オブジェクトはいうに及ばず、半完成オブジェクト、オブジェクトの要素部品であるデータ構造の定義モジュールの完成品やその破片、メソッドモジュールやその破片等様々なものがある。これらを対象世界のクラス階層構造内部にあるいはインスタンスオブジェクトの集合世界として、系統的かつ効率的に格納・管理・検索・取り出し・起動をするための支援環境が欲しいというのがその要求の主旨である。

そこで、我々はオブジェクトベース的なオブジェクト（以下OBオブジェクトと称する）を扱うシステムをオブジェクトベースリポジトリ（OBR）として構築してきた [2]。現在このOBRはリポジション支援環境として開発中であるが、システムの一部は既に稼働を始めており、支援環境としてのOBRの構成や機能もほぼまとまってきている。そこでOBR開発研究の一環として、従来からある類似のシステムであるOODB、ソフトウェアリポジトリ、OB機構 [3, 4] 等との比較・対照を行うことで本OBRの要求仕様を再確認して固めると共に、取り入れるべき特徴・機能、逆にOBRの更に延ばし差別化すべき機能や特性等を考察した。

更に、本OBRは本章の冒頭に出てきたオブジェクトベースシミュレーションの支援環境自体の基盤環境としても期待されている。つまり、上記のOBRの機能・役割は勿論、オブジェクトベース（OB）シミュレーション支援環境全体、言い換えればOBシミュレーション支援環境自身もOBRで格納・管理・起動出来る様なシステムとすることが期待されている。これはソフトウェアリポジトリの機能・役割をも兼ねるものである。このようにOBRはOB機構と共に、OBシミュレーションシステムの基盤機構として、UNIXシステムで言えばファイルシステムに対応する重要な支援環境である。本報告においてはOBRのこの様な将来的な展望と知見を引き出すためにもOBRと他のリポジトリシステムの現在及び将来についても言及する。

2 開発の背景、動機

2.1 オブジェクトベースシミュレーション

我々は流体力学（希薄気体力学）分野といった、工学での分析・解析対象になるような現象世界のシミュレーション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、オブジェクト指向によるモデリングパラダイムが有効である [3, 4]。これは、データと振舞い (操作) が一体である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というものである。我々は、このような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。このオブジェクトベースシミュレーションはオブジェクトをモデル化の単位とすることで、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なうことが出来る。しかし一般の応用分野では、この手法を用いることは殆ん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 [3, 4]」が行な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 (再構成モデリング) 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ほとんどの場合、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。これをある程度、解決するシステムとして

オブジェクトベース機構（以下OB機構）がある[3, 4]。これは、オブジェクトを単位として格納、管理、変更などができるシステムの呼称である。似たものにOODBがあるが、そちらは一般にオブジェクトの静的データ構造は管理するものの、メソッドを管理することが出来ない。一方、我々の研究グループのOB機構は、OODBであるONTOSを基に構築され、静的な面であるデータ構造以外に動的な面であるメソッドの管理をも行なわせることでオブジェクトを単位として取り扱うことを可能としてある。OB機構の目的は、オブジェクト指向モデリングを用いて(高い精度で)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そのままコンピュータ上で管理・起動することである。このOB機構を利用することで、モデリングの際のプログラミングの手間が少なくなり、さらに対象世界の物体をモデル化したオブジェクトがその操作も含めてそのまま管理されるので、従来よりもシミュレーション対象世界の構築がはるかに容易となる。

2.2 統合環境の必要性

OB機構は我々がオブジェクトベースシミュレーションを行なう際に、オブジェクトを管理するシステムとして有効である。しかし、OB機構だけでは実行時の支援が中心になり、従来と異なったモデリング手法であるOBシミュレーションの上流過程をサポートするには、未だ不十分である。換言すれば、OBシミュレーションにおけるモデル化オブジェクトの生成支援を行なうシステムは実動・実用システムとしてはまだ存在しないと言っても良い。実際、OBシミュレーションの生成支援を考慮すると駆動だけでなく、開発段階から一貫した支援システムが期待された[1, 2]。このように、OBシミュレーションでは従来のシミュレーションと手法が異なり、対象世界を全てOB的オブジェクトとして表現することが必須であるので、全体においてまだ支援環境が不十分であると言える。また別の要求として、シミュレーションという目的の性質上、対象をモデル化したオブジェクトを格納・起動し、さらに修正して再格納するというトライ&エラーの過程を繰り返すことが挙げられる。結果として、多様な形態や完成度・構成を持つOBオブジェクトの実現・駆動・管理・運用・永続格納の機能を持った、生成・駆動・表現の各支援環境が期待されることになる。しかし、それらの支援環境を独立に構築しては一貫したモデリングに支障をきたす恐れがある。つまり、これらの支援環境と対象世界の実現プログラムやその他のアプリケーションとOSとのやり取りに共通のインタフェースを持ってを支援・管理するという統合的な支援・管理システムの必要性が高いのは明らかである。

そこで用途をOBシミュレーションに限定し、設計から駆動に至るまでのオブジェクトライフサイクルの管理・永続格納などを行なう、OBシミュレーション及びその支援環境群を支援する共通基盤機構を構想した。このようなシステムはまず当初はシミュレーションを行なうためのプロダクト（つまりオブジェクト）を管理するという側面を持つ点から言ってその一部の側面はソフトウェア工学的リポジトリ[5, 6, 7]に近いものとして位置付けられる。ここで基本概念でもある「オブジェクトベース」を考慮して、これをオブジェクトベーストリポジトリ（OBR:ObjectBased Repository）と名付けた。

3 概念設計

3.1 OBRの定義

ここで、OBRを明確に定義する。

> オブジェクトの設計・生成から、テスト駆動・改訂・再テストやシミュレーション駆動・データ記録・ハンドリングに至る、OBオブジェクトライフサイクルの全過程をリポジションサポートすることが可能なリポジトリシステムであり、中でも本研究では特にシミュレーション目的の比較的小規模の、かつチームせいぜい数人を対象とした、オブジェクト管理支援環境を指す。

ここで、規模をチームせいぜい数人に限定しているのは、目的であるシミュレーションではこのくらいで十分であると考えたからである。

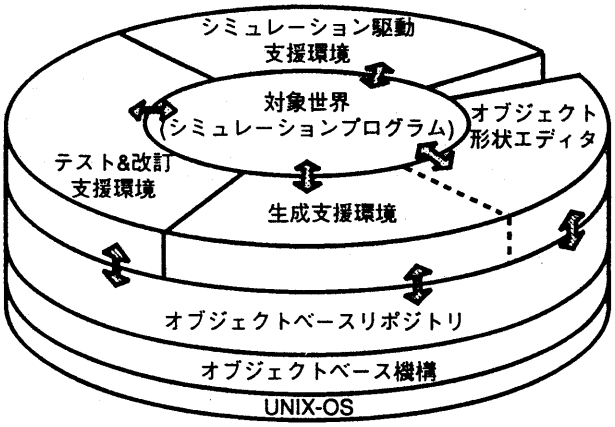

図 1: オブジェクトベースシミュレーション支援環境全体概念図

3.2 OBR の支援機能

OBR を用いたシミュレーションは対象世界に対するモデリング作業過程を支援するシステムとして、以下の 3 つの支援環境によってサポートされる (図 1)。

1. 生成支援環境
   シミュレーションの対象世界をオブジェクトとしてモデル化して、オブジェクトの性質の定義と属性の設定をし、実際に OB オブジェクトを生成する。本支援環境にはオブジェクトの幾何形状を GUI で直接生成する形状エディタを含む。

2. テスト＆改訂支援環境
   生成したオブジェクトに対し、シミュレーションのテストを行ない、改訂が必要なら前段階に戻る。

3. シミュレーション駆動支援環境
   オブジェクトの仕様を固定し、高効率・本格的なシミュレーションを行なう。

OBR はこの 3 つの支援環境及び対象世界オブジェクトを以下の様に支援する。

- 生成支援環境

  1. シミュレーションの対象となるモデルをオブジェクトとして定義する。
     このとき、定義のためのクラスは OBR で用意する。
  2. 定義されたクラスのインスタンスを生成する。

  これらのクラス情報やインスタンスは OBR が管理・格納する。

- テスト＆改訂支援環境

  1. 生成されたインスタンスに対し、シミュレーションのテスト駆動を行なう。単位オブジェクトの内部及びオブジェクト間の相互関連リンクはまだテンポラリ状態のままであり動的に変更可能 (非固定的) である。相互関連リンクは OBR で管理する。
  2. シミュレーションの初期データや結果は OBR が管理・格納する。
  3. シミュレーション結果に応じて、クラス定義やインスタンスの属性値を変更する。

  クラス定義やシミュレーション駆動結果などオブジェクトに複数のバージョンがある場合、OBR がそれらのバージョン管理を行なう。

- シミュレーション駆動支援環境

  1. クラス定義を固定する。ここで、リンク構造を動的（テンポラリリンク）から静的（固定的モジュール構成）に変更する。
  2. 本格的なシミュレーションを行ない、データを得る。

  テスト&改訂支援の時に較べてリンク構造が静的になっているので、高速化が期待できる。

これをふまえて、OBR に最小限必要だと思われる機能を次のように設定した。

1. オブジェクトライフサイクルの段階に関わらず、オブジェクトの格納・管理・検索・取り出しを行なう機能。
2. オブジェクトの定義のためのメタデータ管理機能。
3. オブジェクト間の関連（リンク）を管理する機能。
4. データファイルやアプリケーションプログラムを管理する機能。
5. アプリケーションの起動を管理 (あるいは支援) する機能。

3.3 OB 機構上の OBR

OBR と OB 機構は機能的な面から見ると類似点が多い。そこで、概念設計時点での OB 機構から OBR を支援可能な点を挙げる。まず、OB 機構は OB 的オブジェクトを扱うという点で OBR と共通である。そのオブジェクトに対する処理としても格納・管理・検索・取り出し・起動を行なう機能を備えている。また、駆動支援が目的である OB 機構は OB シミュレーション駆動データを管理する。これらの機能は OBR の設計に含まれるものであるので、OB 機構から支援可能である。

一方、新たに作成する必要があるのは

- ライフサイクルを問わない管理機能
- メタデータ管理機能
- オブジェクト間の関連を管理する機能
- ファイルを管理する機能

といったところが考えられている。

4 OBR のプロトタイプの実装

OBR は基本となるシステムとして OODB である ONTOS[8] を用いているので、ONTOS が動作している UNIX システム (HP9000/ 715, HP-UX9.05) 上で構築した。

4.1 プロトタイプの実装方針

システムの実装に関して以下のような方針を立てた。

1. プログラミング言語は C++を用いる。
2. 基本システムは OODB を利用する。
3. クラス階層を用いて対象世界を表現する。
4. 対象世界に依存する部分は、ユーザが設定できるようにする。
5. アプリケーションは OBR と独立した動作も可能とする。
6. クライアント-サーバ・アーキテクチャを用いる。

4.2 基本的な機構

これらと、前述の OBR の最小限の機能を考慮して次のような機構の実装が必要であると考えた。

- ONTOS を利用した永続的オブジェクト管理機構。
- モデル化されたオブジェクトをソースレベルで管理する機構。
- メタレベルのクラスやインスタンスを管理する機構。
- モデル化されたオブジェクトの属性・メソッドの動的付与機構。

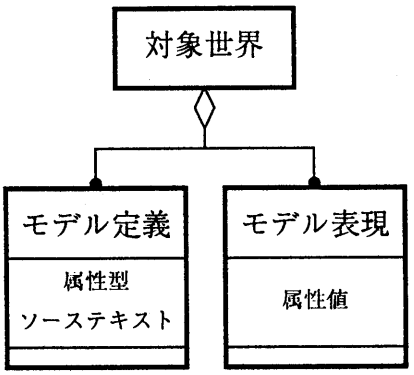

図 2: 対象世界の表現

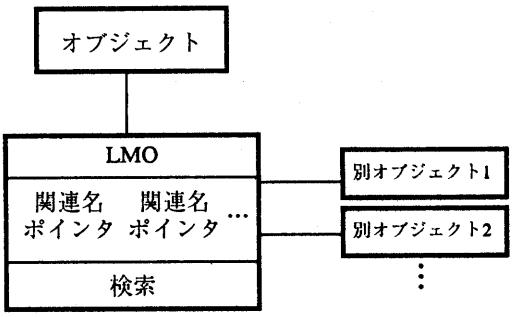

図 3: リンク管理オブジェクト

- オブジェクト間の関連を記憶・管理する機構。
- ファイルをオブジェクト同様に扱う機構。

4.3 各機構の実装

永続的オブジェクト管理機構

前述のように、OBR の基本システムには OODB-ONTOS を用いる。従って ONTOS でサポートされている機能が利用できる。この機構は、ONTOS でサポートされている「永続的インスタンス管理」機能と OB 機構で構築された「メソッド管理」機能を組み合わせ、データとメソッドを 1 セットとした OB オブジェクトの「永続的オブジェクト管理」を実装したものである。

ソースレベルの管理

モデル定義クラス [2] はメタクラス的な役割を果たす。ユーザが定義するクラスの情報は、モデル定義クラスのインスタンスとして管理される。また、その際のバージョン管理についての簡便性を考え、メソッド単位での管理機能を持たせた。

メタレベルのクラスとインスタンス

C++のクラスの集約階層を用いて対象世界を表現した（図 2）。表現法は OMT である。対象世界の様々なクラスはモデル定義クラスのインスタンスとして定義される。ここで定義されたクラスのインスタンスはモデル表現クラスのインスタンスとして管理される（図 2 参照）。モデル表現クラスは「モデル定義クラスで定義されたクラス」の「インスタンスとしての属性」を管理するためのクラスである。ONTOS の永続的オブジェクトとは別に「インスタンスとしての属性」のみを管理することで、クラス定義に変更があってもそれまでに生成したインスタンスで矛盾がないものは消去しないでおく。

属性・メソッドの動的付与

対象世界のオブジェクトの定義と属性は「モデル定義クラス」と「モデル表現クラス」のインスタンスとして動的に格納管理される。このため、属性が保存された上でクラス定義を変更することは属性・メソッドの動的付与（削除含む）を意味する。

オブジェクト間のリンク管理機構

汎化構造・集約構造・ネットワーク構造・グループ構造などを含めた多種多様なオブジェクト間の関連を管理し、動的な組合せ変更を可能にするためには、「関係及び関連」を記述できるオブジェクトを用意する方法が良いと考えた。このオブジェクト間にリンクを張るオブジェクトをリンク管理オブジェクト（LMO:Link Management Object）と名付けた（図 3）。必要だと思われる情報として「関係の内容」「関連を持つオブジェクトのテーブル」がある。同

様に必要な機能は「オブジェクト間の関連を検索する機能」と「関係からオブジェクトを検索する機能」が挙げられる。この機能が無ければ関連を管理することは不可能である。

ファイル管理オブジェクト

OBRでは「コンパイラ」などの外部のアプリケーションの起動機能が必要である。このようにUNIXファイルを管理する目的で管理対象となるファイルと一対一対応のファイル管理オブジェクト (FMO:File Management Object) を実装した。ただし現在のところ、UNIXファイル自体にトランザクションの概念を用いてはいない。この点は改良する予定である。

4.4 現在の実装状況

前述の様々な機能を実現する「モデル定義クラス」「モデル表現クラス」「LMO」「FMO」などの主なクラスのうち、「LMO」「FMO」は既に実装が終っているが、メタレベルを扱うクラスは実装が未だ不完全である。

5 他のリポジション支援システムとの比較

OODB (ONTOS)・OB機構・ソフトウェアリポジトリ (PCTE)・OBRについて比較・考察する。

5.1 データモデル

OODB

OODBのデータモデルはオブジェクト指向データモデルである。つまり、「オブジェクトの属性、データ構造」を扱う単位とし、その操作はデータ構造の定義と同時に決定される。通常はクラス定義やメソッドはOODB外で別途管理する場合が多い。

OB機構

OB機構のデータモデルは、オブジェクトベースモデルとでも言うべきものである。即ち、「オブジェクト指向データモデル＋メソッドの管理＋強固なカプセル化機能」となる。OODBではオブジェクトのデータ構造を扱う単位であるが、OB機構では「オブジェクト＝データ構造＋メソッド＋相互関係＋OBインタフェース」を単位としている。

ソフトウェアリポジトリ (PCTE)

PCTE[5]のデータモデルは、「オブジェクトベース」のモデルとスキーマ定義集合 (SDS) である。この「オブジェクトベース」は我々のいうところのオブジェクトベースとは異なる意味合いである。この場合のオブジェクトは「オブジェクト＝データ構造＋リンク」であり、その汎用性からデータに対する操作はツール依存となっている。また、型の定義に関連してそのデータモデル自身がSDSで表現される。

OBR

OBRのデータモデルはOB機構の「オブジェクトベースデータモデル」にデータ構造として「単位オブジェクト内及びオブジェクト間の関連」が加えられたものである。

これらを比較すると、OODBではデータの静的管理にオブジェクトが構造化されているという事実を利用し、操作が付随している。つまり処理の単位をオブジェクトのデータ部に限定し、オブジェクト指向の柔軟なデータ表現を活用して多様なデータが扱えるという特徴がある。OB機構は処理の単位をデータ部だけでなくメソッドも含めたオブジェクト自体にしている。このときクラス間の相互関連はONTOSの機能をそのまま利用しているので新しい相互関連の型の定義はサポートしていない。OBRはこの部分をシステムがサポートした形になっている。PCTEは結果的にOBRのモデルと似たものになっているが、OBRの型定義は用途を限定していることによって、SDSのように高い汎用性を持ってはいない。

5.2 機能の比較

OODB

OODBの機能の特徴はオブジェクトのデータ構造を用いることで、「階層構造や不定長データが扱える」「インスタンス間の動的な関連付

けができる」「新しいデータ型を定義できる」といったことが挙げられる。

OB 機構

OODB と比較すると、「新しい操作を動的に定義できる」「対話的言語によりメソッドを管理できる」「オブジェクトベースシミュレーションの駆動支援」といった新たな機能が増えている。

ソフトウェアリポジトリ（PCTE）

PCTE は標準リポジトリとしてのツールインターフェースの定義であるので、機能の比較という意味ではあまり意味がないと言えよう。

OBR

OBR の場合、機能的には OB 機構をさらに押し進めたものとなっていて「クラス間の動的な関連付け」「メソッドや属性を含めたオブジェクトのバージョン管理」「生成途上オブジェクトを駆動することが出来る」というような機能の追加がある。

OODB から OBR に至るまでの機能の追加は、データベースの扱うデータをオブジェクトに拡張し、従来の OODB でデータに対して行なえる操作をオブジェクトに対して行なう形になっている。換言すれば、多様なオブジェクトを統一的に管理するシステムであるとも言える。このため、柔軟なビューの変更などが容易になっている。一方、ファイルに対するアクセスが増えたため、排他制御やトランザクション制御が不備な点として挙げられる。

6 おわりに

オブジェクトを管理するシステムとしての OBR をオブジェクト指向データベースの発展形システムとして用いた場合の考察をした。将来的には、オブジェクトを単位とした処理がデータベース同様に扱えることが期待でき、OB シミュレーション支援環境の共通基盤機構として高度な統合環境が構築できると思われる。また、今後の課題として「オブジェクト間の関連に対する言語的な問い合わせ機能」「ファイル管理も含めたトランザクション機能」などが挙げられる。

参考文献

[1] 加藤木和夫, 畠山正行:「オブジェクト指向シミュレーション・モデル記述の開発支援環境」、第 98 回情報処理学会ソフトウェア工学研究会研究報告、Vol.94,No.43,pp.9～16、1994 年 5 月 25 日.

[2] 加藤木和夫, 畠山正行, 小林秀行:「オブジェクトベース・リポジトリを用いたオブジェクト生成支援環境」、第 101 回情報処理学会ソフトウェア工学研究会研究報告、Vol.96,No.44,pp.9～16、1994 年 11 月 18 日.

[3] 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.

[4] 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 回ハイパフォーマンスコンピューティング研究会報告、Vol.94,No.51,pp.1～8、1994 年 6 月 17 日.

[5] Lois Wakeman & Jonathan Jowett 著 PIMB 協会 編 松本吉弘 監訳:「PCTE 開放型リポジトリのための標準 - ソフトウェア利用環境の基盤 -」、日科技連株式会社、1994 年.

[6] 落水浩一郎:「ソフトウェア・レポジトリ」、情報処理 Vol.35 No.2(Feb.1994)、pp.140-149.

[7] 堀内 一:「ソフトウェア環境におけるリポジトリへの要求 - - リポジトリ内容標準化の要件 - -」、情報処理学会研究報告,**94-se-101**(1994.11.18)、pp.65-72.

[8] ONTOS DB3.0 マニュアル、ONTOS Inc. 1995.