# ビジネスオブジェクト制御フレームワーク

## Business Object Control Framework

北川達朗, 遠藤英幸

**要　約**　CORBAをはじめとする分散オブジェクト環境が徐々に広まるにつれ，その問題点も指摘されはじめている．オブジェクト認識と表現方法の相違に起因するオブジェクトの再利用性の眼界である．その問題に対する回答として，コンポーネントモデルが提言され，いくつかの実装プロダクトも市場に出始めている．OMGにおいても，CORBA層の上位にコンポーネントモデルを実装しビジネスオブジェクトの標準化に着手した．ビジネスオブジェクトの再利用性を高いレベルで実現することを狙ったものである．

本稿では，基本的にはOMG/BODTFの規定に沿いつつ，汎用的なビジネスオブジェクト制御のためのフレームワークの実現案の一つを提案する．

**Abstract**　The steadily growing popularity of distributed object environments like CORBA has made it necessary to focus on the problems in such systems. One such problem is the limitation of object reusability due to the differences in object recognition and representations.

The Component Model has been suggested as the answer to that problem and already implemented in some products that have appeared in the market. For the Object Management Group (OMG), the implementation of the Component Model above the CORBA layer has been started in order to standardize the business object.

This is a trend toward realizing a business object which offers high-level reusability.

This paper suggests one implementation idea of the framework for a general-purpose business object control, while complying with the OMG/Business Object Domain Task Force (BODTF) definition.

## 1. はじめに

オブジェクト指向によるソフトウェア開発は，オブジェクト技術や各種プラットフォーム技術の普及によって，徐々に浸透しつつある．業務(あるいは問題)を分析し，その解決のためのシステムをオブジェクトベースで設計・開発し，システム化を行うことが現実に可能となりつつあり，オブジェクト指向開発ツールを用いた業務パッケージの商品化も盛んになってきた．オブジェクト指向の目的とするところは，言うまでもなくソフトウェアの部品化と再利用性の確保であり，その成果としての開発生産性向上・保守効率の向上にある．しかし，現実には，その効果が十分に現れているとは言い難いであろう．その理由としては

- 各々のシステム・パッケージが，個別の方法論やオブジェクト認識に基づいている
- 開発言語と実行環境に深い依存関係がある

といったことがあげられよう．これらのことが，オブジェクト指向の長所を矮小化し，オブジェクトベースのシステム化に対する大きな弊害となっている．各々のシステム・パッケージにおいて，オブジェクトの表現方法も違えば，実装方法や開発ツール

も異なる状況では，使用者にとって不便きわまりないし，オブジェクトの再利用性もごく限定された範囲に留まってしまう．これが，大規模な業務システムをオブジェクトベースで開発することに対する阻害要因となっていたと言えよう．

そこで，オブジェクトの上位に‘コンポーネント’という概念を導入することにより，これらの問題を解決しようという考え方が広まりつつある．

本稿では，‘コンポーネント’の概念定義を行い，その有用性を確認した上で，コンポーネントベースのシステム化を推進するためのフレームワーク（ミドルウェア）の規定を試みることにする．

## 2. オブジェクトとコンポーネント

まず，オブジェクトとコンポーネントの各々の一般的な定義の確認を行う．

### 2.1 オブジェクト

‘オブジェクト’とは，『データ（属性または変数）とともにその手続き（メソッドまたはサービス）と意味上の制約（規則やポリシー）を一体化したもの』であり，三つのカテゴリがあると考えられる．すなわち，‘ビジネスオブジェクト’，‘テクノロジオブジェクト’，‘アプリケーションオブジェクト’である．

‘ビジネスオブジェクト’は，ソフトウェアビルディングブロックとして存在するのではなく，ビジネスとその基盤としての情報システムをビジネス用語によって理解することを可能とするオブジェクトである．OMG（Object Management Group）のBODTF（Business Object Domain Task Force）では，‘ビジネスオブジェクト’を『ビジネス領域で活性化されたものの表象であり，少なくともそのビジネス名と定義，属性，振る舞い，関連，および制約を含むものである．その表現は，自然言語，モデリング言語，あるいはプログラミング言語の形態をとる』と規定している．

‘テクノロジオブジェクト’は，プログラミングないし情報技術の概念を表し，アプリケーションに対してシステム基盤および種々のサービス群を提供するためのビルディングブロックとなる．具体例としては，GUI 構成要素，通信基盤，データベース基盤およびアプリケーションフレームワーク等がある．

‘アプリケーションオブジェクト’は，情報を表現し，人間との相互作用を管理し，情報を処理するプログラムのことである．具体的には，‘アプリケーションオブジェクト’は，‘ビジネスオブジェクト’と‘テクノロジオブジェクト’とから組み立てられ，必要に応じてプログラムコードで接着され，アプリケーション固有の仕事を遂行する．

図 1 に示す通り三つのカテゴリのオブジェクトは 3 階層を構成している．

### 2.2 コンポーネント

一口に‘コンポーネント’といっても，人によって受け取り方はさまざまであろう．ある人々にとっては，OLE コントロールなどの GUI を意味し，別の人々にとっては，ビジネスオブジェクトそのものを指しているかも知れない．ここでは，まず二つの代表的なコンポーネントを整理してみる．

1） Java Beans

Java Beans は，Java のコンポーネントモデル仕様であり，Java プログラムの

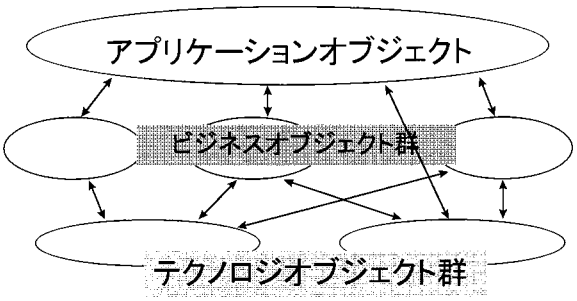


**図 1　3種類のオブジェクトの関連**

ための再利用可能な部品群である．ビジュアル開発環境の上で，個々の Beans を組み合わせるだけで，Java アプリケーション/アプレットが開発可能となる．

Beans の基本機能として低レベルの Java API を隠蔽して，各 API の抽象度を高めることによって汎用的な枠組みを提供する．この中で，イントロスペクション機能と呼ばれる実行時における完全な自己記述のための機構により Beans のプロパティやメソッド，イベントをユーザに公開でき，コンポーネント同士の結び付けを可能としている．また，Beans は自分自身の属性をカスタマイズするための機能や，状態を保存する永続性の機能を持つ．

Java Beans の規約を拡張して，サーバプログラムを，Beans を組み合わせて開発するために必要な機能を追加したコンポーネントフレームワークが，EJB（Enterprise Java Beans）である．これにより，サーバ側にて基幹業務アプリケーションを開発することが可能となる．図 2 に EJB のフレームワークを示す．

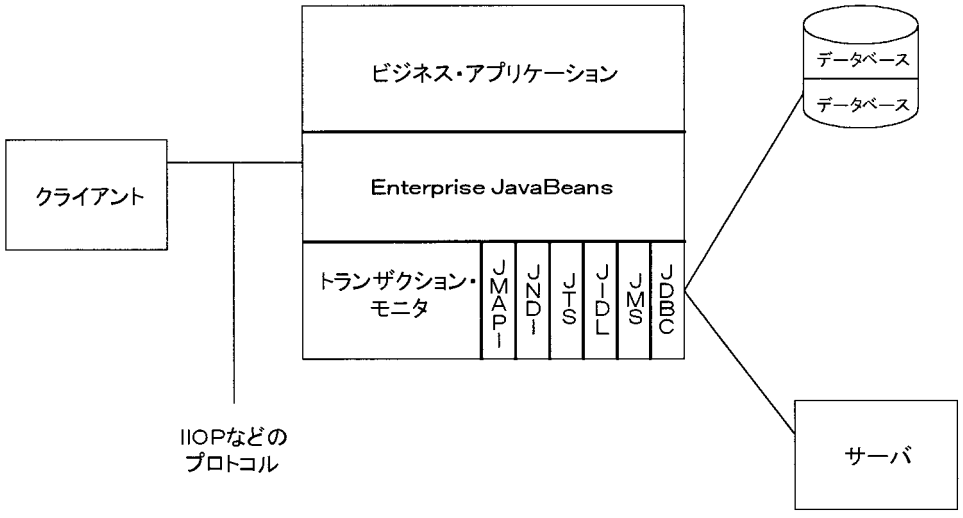


**図 2　Java for the Enterprise のフレームワーク**（出典：月刊ジャバワールド）

2）　DCOM/ActiveX

ActiveX とは，インターネットに対応した OLE（Object Linking and embed-

ding）技術を意味する．DCOM とは，COM（Component Object Model）を分散環境に対応させた技術である．DCOM/COM は，実行プログラムのバイナリモジュールそのものであり，そのバイナリモジュールをソースコードなしに再利用し，システムを構成・再構成することを可能とするための技術であり，ActiveX の中核をなす．つまり，流通しているパッケージ化されたソフトウェア部品を流用してアプリケーションを迅速に組み立てるための方法論とツールを提供する．機能的には以下のものが中心となる．

- コンポーネント間の相互作用と依存性を表現する機構
- 実行時の自己記述のための機構（イントロスペクション）
- コンポーネント間および環境と相互通信するためのスクリプト言語との統合

また，DCOM では，システムオブジェクトという共通機能提供のための実装オブジェクトが存在し，以下の機能を提供する．

- 統一データ転送
  クライアントとサーバ間での双方向データ転送サービス
- パーシステントストレージ
  クライアントとサーバ間で共通に利用可能な永続記憶領域サービス
- モニカー
  オブジェクトに対するネーミングサービス

図 3 に DCOM と ActiveX の関連を示す．

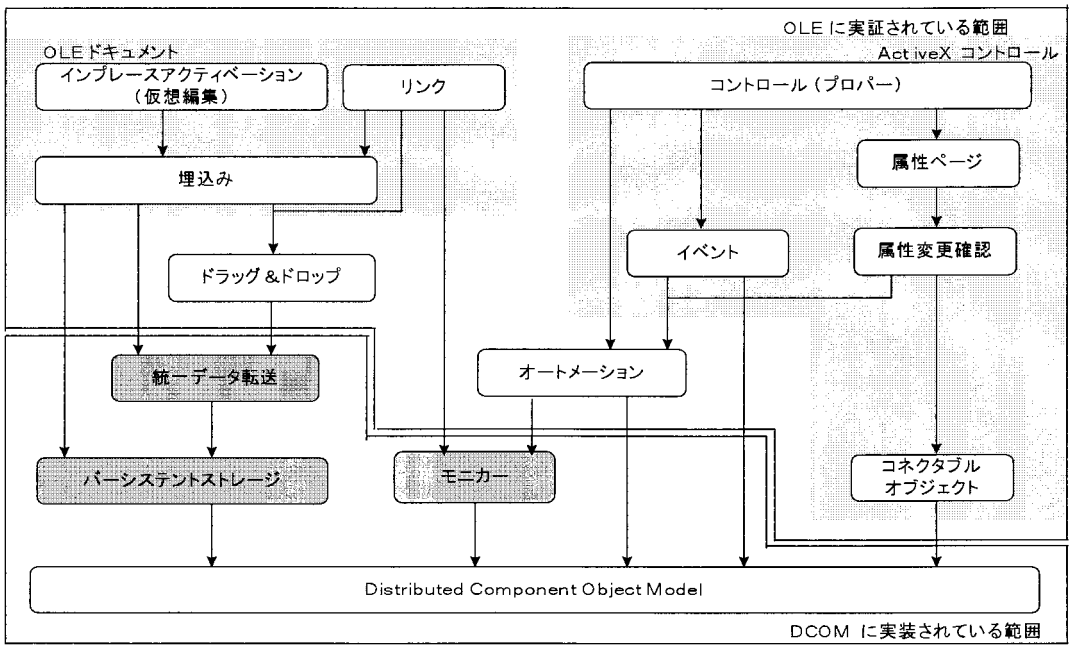


**図 3　DCOM と Active X の関連**（出典：**DCOM** ガイドブック）

以上，代表的な二つのコンポーネントを比較すると，コンポーネントの必要条件が浮かびあがってくる．それは，機能としてのイントロスペクションであり，実装技術としての共通オブジェクトサービスである．

### 2.3 CORBA コンポーネント

さて，ここで中心テーマである CORBA におけるコンポーネントに話を移す．OMG では，ビジネスオブジェクト領域に対する標準化に乗り出し，BODTF というタスクフォースを設立し，共通フレームワークとプラットフォームの制定に対して作業を開始している．それまでの CORBA 技術とは，『インタフェース定義と管理』にすぎなかったが，コンポーネント技術の成果を取り入れて，効率的な CORBA アプリケーションの作成のためのコンポーネントモデルの作成を目指している．このモデルを BOF（Business Object Facility）という．BOF は，BOCA（Business Object Component Architecture）に引き継がれ，現在規格化に向けての作業が行われている．

本稿では，コンポーネントモデルを OMG/CORBA プラットフォーム上における

『オブジェクト指向のソフトウェア要素の組み立て・実行のためのメカニズム』

と位置づけ，ビジネスオブジェクトの再利用と，そのビジネスオブジェクトの制御のためのフレームワーク（ミドルウェア）がもつべき要件を 3 章以下に記述する．

なお，OMG では，前述の BODTF の他に，ORBOS（Object Request Broker and Object Service）という，より CORBA 基盤に近い部分を取り扱うタスクフォースにおいても，コンポーネントを取り扱っている．両者の棲み分けは必ずしも明確ではないが，ビジネス領域を取り扱うという観点からここでは，BODTF にて取り扱われている高度な（上位層に位置付けられる）コンポーネントを対象とする．

## 3. ビジネスオブジェクト制御フレームワーク

### 3.1 分散オブジェクトベースの開発

従来，CORBA 分散オブジェクト環境において，ビジネスアプリケーションは，IDL（Interface Definition Language）による形式的なインタフェースの記述によって設計され，それに従った開発が行なわれてきた．しかし，この方法では，

- オブジェクト分析結果の表現方法とオブジェクトへのマッピング手法が均一でない
- オブジェクト設計において，そのオブジェクトが担うべき実際の役割を十分に表現することはできず，設計者の意図が十分に実現できない

という状況が往々にして発生する．この状況を改善し，より厳密にアプリケーションやその部品としてのビジネスオブジェクトを定義し，オブジェクトとしての実装に結び付けるために，設計・開発の各段階において，「ビジネスセマンティックな情報」を付加する仕組みを組み入れるとともに，実行時にそれを制御する機構を構築することを考える．これを具現化するためのフレームワーク（ミドルウェア）として，ビジネスオブジェクト制御フレームワーク（以下，BOCF と記述）を位置づける．つまり，前述の BOF（BOCA）実装とそれに付随する機能群を BOCF として位置づけることになる．

図 4 の左は，現状の CORBA 分散オブジェクトの階層構造を示す．分散オブジェクト基盤上に直接アプリケーションオブジェクト，ビジネスオブジェクトが構築される．ビジネスセマンティックな情報は，ユーザがオブジェクトのメソッドとして直接コーディングする必要がある．

一方，図 4 の右は，BOCF をミドルウェアとして実装した形態を示す．セマンティック制御層として BOCF が介在することにより，ユーザプログラムはその部分のコーディング記述から解放されることになる．

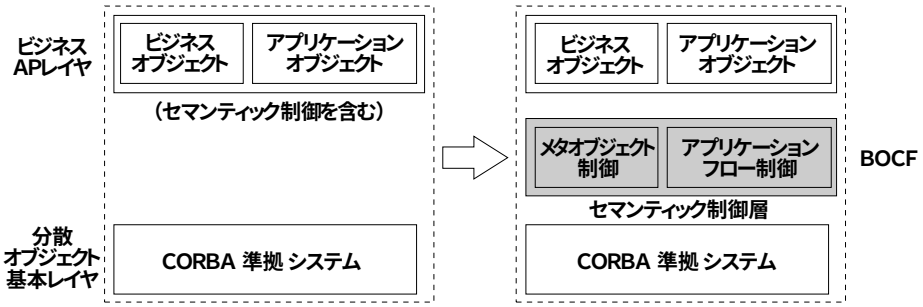


**図 4　BOCF の位置づけ**

### 3.2　理想的な分散オブジェクトの開発・実行環境

BOCF は，理想的な分散オブジェクト環境の要件として，以下の 3 点を重視する．

- ビジネスオブジェクト・アプリケーションオブジェクトの再利用性の実現
- 既存のシステムリソースの有効利用の実現
- 異種の情報技術の柔軟な取り込みを可能とする柔構造の実行環境の提供

本稿では，特に第一の要件に焦点を当てて記述する．

ビジネスオブジェクト，アプリケーションオブジェクトの再利用性の確保には

1）　既存オブジェクトの流通

2）　既存オブジェクトのカスタマイズ

の二つの側面を考慮する必要がある．既存オブジェクトの流通とは，一度作成したビジネスオブジェクトの属性（プロパティ・メソッド・イベント等）を公開し，第 3 者がそれを適用可能とするための仕組みを作ることである．2.2 節で述べた Java Beans や DCOM におけるイントロスペクション機能に相当する機能に，オブジェクトの登録・削除，リポジトリ運用，ソフトウェアの配布，課金等の機能が必要となる．

他方，既存オブジェクトのカスタマイズであるが，アプリケーションオブジェクトにしろ，それを構成する部品としてのビジネスオブジェクトにしろ，既存のものを全く変更しないで利用可能なケースは稀である．したがって，オブジェクトにおいて，そのオブジェクトを特徴づける固定部分と，変更したり選択したりすることが可能な可変部分が明確化され，その枠組みの中で正当にカスタマイズできるかどうかが，そのオブジェクトの再利用性を大きく左右することになる．そして，そのカスタマイズのための開発方法論の提供と，カスタマイズしたオブジェクトの実行環境の提供が BOCF にとっての重要課題となる．各々のオブジェクトの再利用性の意味合いは以

下のようになる．

1)　アプリケーションオブジェクトの再利用性

使用者が，対象業務のオブジェクト分析を行う際に，典型的業務プロセスが予めモデルとして提供され，その分析結果を流用することを可能とする．これにより，一から分析を開始する困難さ・煩雑さを回避することが可能となり，特に開発の上流工程におけるアプリケーション開発の生産性は大きく向上する．

2)　ビジネスオブジェクトの再利用性

実際のアプリケーション構築に際して，適用する個々のビジネスオブジェクトが持つ属性・メソッド等をカスタマイズする手段を提供することにより，ビジネスオブジェクトの開発量を劇的に削減することが可能となる．

上記のオブジェクトの再利用性を実現するテクノロジオブジェクトとして，BOCFを位置づける．そして，BOCFがアプリケーションオブジェクトおよびビジネスオブジェクトの開発環境・実行環境を提供する．

## 3.3　セマンティック制御機能

アプリケーションオブジェクトやビジネスオブジェクトを再利用可能とし協調動作させるためには，ビジネスセマンティックを制御することが必要となる．

図5は，セマンティック制御層としてのBOCFを示している．業務のオブジェクト分析の結果抽出されるビジネスオブジェクトモデルを，UML（Universal Modeling Language）で表現する．UMLは，標準のモデリング言語としてOMGで規格化されており，過去の種々のオブジェクト分析・設計の集大成とも言える位置づけのものである．UMLで表現した業務分析結果を，最終的にはIDLに落とすことになるが，その間に，セマンティック制御層特有の処理が介在する．その処理系に対する入力としてSDL（Semantic Definition Language）を定義する．SDLは，概ね，これもOMGで規格化されているCDL（Component Definition Language）に相当するであろう．ただし，CDLの記述能力でセマンティック表現が十分に実現できるか否かが不明であるため，本稿ではあえてSDLという新規の言語を使用する．

BOCFは，上述のセマンティック制御のための開発環境・実行環境を提供する．

以下の節で，開発・実行の各環境を概観する．

## 3.4　開発環境の提供

オブジェクト指向によるシステム開発は，概念的に図6のようなサイクルを持つと考えられる．（便宜上，保守フェーズを除いている）

BOCF開発環境は，図6の概念を具体化し，ユーザに対して，アプリケーションオブジェクト/ビジネスオブジェクトの開発を，容易に実現するための枠組みと，それを支援する種々の機能を提供する．

### 3.4.1　アプリケーション開発の流れ

BOCF開発環境による開発は，以下のフェーズに分かれる．

1)　業務分析フェーズ

上流CASEツールを使用して業務分析を行う．分析手法は，既存のオブジェクト分析手法をとるものとする．業務分析結果を，UMLによって表現する．表現形態は，UMLが規定する図のうち

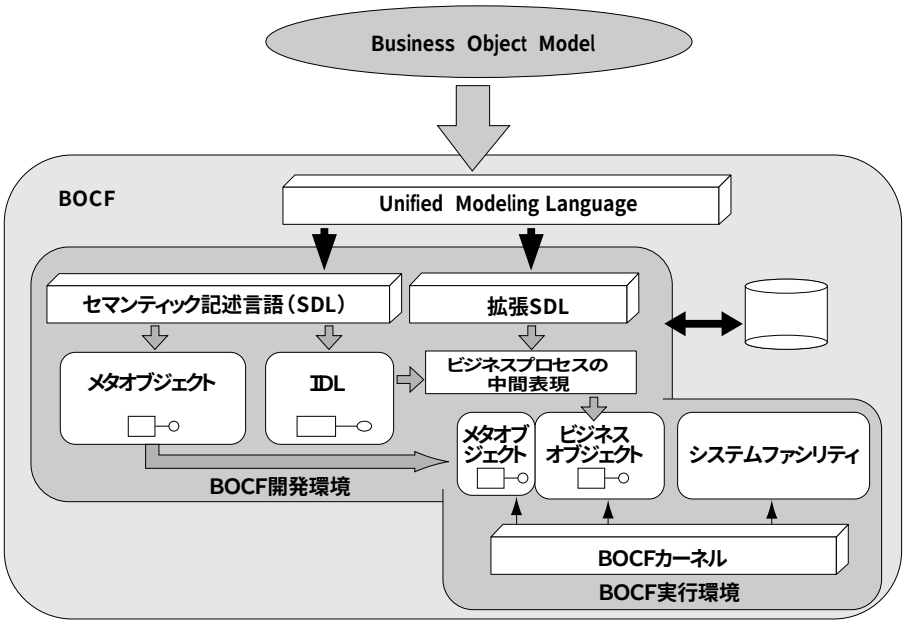


**図 5　セマンティック制御層としての BOCF**

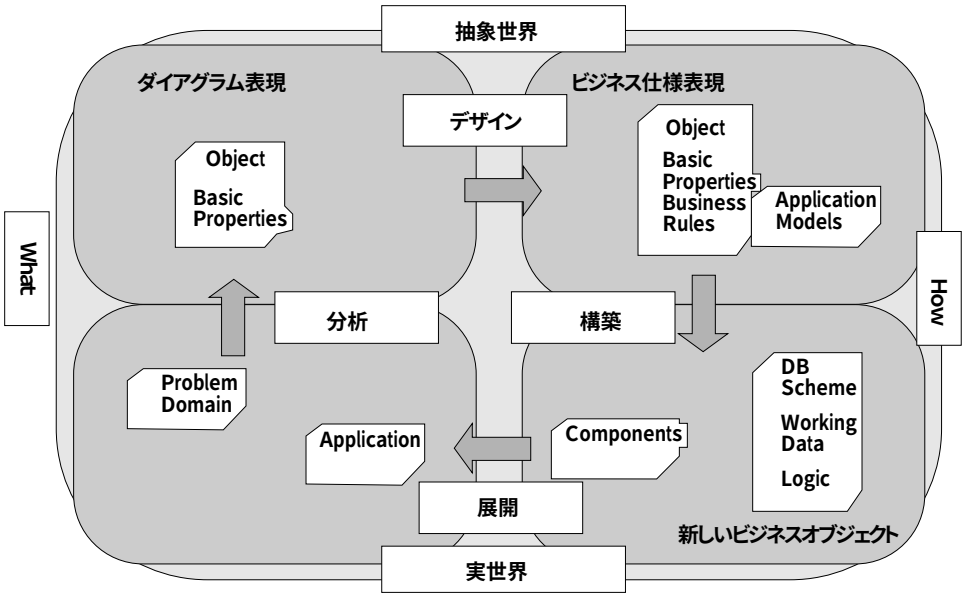


**図 6　開発サイクルの概念**

・クラス図（Static Structure Diagram）
・クラス状態遷移図（State Diagram）
・シーケンシャルダイアグラム（Sequence Diagram）

の形により表現することとする．必要に応じて，図の追加を行う．

2） 開発フェーズ（アプリケーションオブジェクト，ビジネスオブジェクト，メタ

オブジェクトの生成）

アプリケーションを構成する部品として，すべて既存のビジネスオブジェクトが使用可能な場合は，アプリケーションオブジェクトの作成のみとなる．UML にて表現された上記 3 種類の情報をコンパイルし，その結果をもとに，『ビジネスプロセスの中間表現』が生成される．これは，アプリケーションオブジェクトのメソッドの自動生成に相当する．その成果物と，既存のビジネスオブジェクトに対応するメタオブジェクトと IDL の情報を連結して，アプリケーションが構築される．ビジネスオブジェクトを新規に開発が必要な場合は，以下の手順によりビジネスオブジェクトを作成した後，上のアプリケーションオブジェクトの作成を行う．

### 3.4.2 ビジネスオブジェクトの開発手順

ビジネスオブジェクトの開発手順は，以下のようになる．

1) ビジネスオブジェクトとすべき業務のオブジェクト分析を実施し，クラス図/クラス状態遷移図と，ビジネスロジックをシーケンシャルダイアグラムとして記述する．

2) クラス図/クラス状態遷移図を UML コンパイルし，その結果をもとに SDL にてセマンティック表現を行い，それを SDL コンパイルする．利用可能なビジネスオブジェクトがあれば，そのセマンティック情報をアプリケーションに組み込む．その結果として，IDL ソースとメタオブジェクトが生成される．

3) シーケンシャルダイアグラムを UML コンパイルし，その結果をもとに，『ビジネスロジックの中間表現』を作成する．

4) 開発者は SDL で定義できない動的な部分をビジネスオブジェクトのメソッドとしてプログラミングし，3）で生成した『ビジネスロジックの中間表現』と連結する．

5) IDL ソースを IDL コンパイルし，CORBA のスタブ&スケルトンを生成し，『ビジネスロジックの中間表現』と連結して，ビジネスオブジェクトを生成する．上記の『ビジネスプロセスの中間表現』，『ビジネスロジックの中間表現』とは，メソッドの表現形式を意味する．

### 3.4.3 オブジェクトの再利用性

上記の手順で開発工程が終了するわけであるが，開発したアプリケーションオブジェクト/ビジネスオブジェクトの再利用の観点からみると，以下のようになる．

1) アプリケーションオブジェクトの再利用

『アプリケーションを表現しているビジネスプロセスの再利用』を意味する．

利用形態は以下の通りとなる．

① アプリケーションオブジェクトをそのままの形で再利用する

リポジトリに格納されているアプリケーション，およびそれが使用するビジネスオブジェクト群を選択するだけでよい．

② アプリケーションをカスタマイズして再利用する

・アプリケーション・モデルから，構築すべきアプリケーションのビジネスフローに近いものを選択する．

・ビジネスプロセスの変更，プロセスを構成するビジネスオブジェクトの変更等のカスタマイズを行う
・アプリケーションの構築を行う．ビジネスオブジェクトのカスタマイズを行った場合は，ビジネスオオブジェトの生成を行った後，AP の構築を行う．

2） ビジネスオブジェクトの再利用

『ビジネスオブジェクトのメソッド/属性の再利用』を意味する．利用形態は以下の通りとなる．

① ビジネスオブジェクトをそのままのかたちで再利用する．

あらたにアプリケーション（ビジネスプロセス）を定義し，そのビジネスプロセスを構成する部品として，既存のビジネスオブジェクトおよびメタオブジェクトを使用する．

② ビジネスオブジェクトのメソッド/属性をカスタマイズして再利用する．

構築するアプリケーションのニーズにあわせた，メソッド/属性の変更を行う．変更は，SDL に反映される．その SDL をコンパイルし，ビジネスオブジェクトおよびメタオブジェクトを再生成した後, 対象のアプリケーションを再構築する．

### 3.4.4 統合開発環境

BOCF 統合開発環境として，図 7 のような環境が考えられる．
ビジネスオブジェクトワークベンチにて，実現する機能は以下の通りである．

- UML 利用環境の提供
- UML 継承ツール（SDL 出力）の提供
- メイクツールの提供
  - ・メタオブジェクト生成ツール（SDL コンパイラ）
  - ・アプリケーションオブジェクト/ビジネスオブジェクト生成ツール
- IDL 利用環境の提供
  - ・基盤とする CORBA プロダクト毎の IDL 利用環境
  - ・メソッド記述言語による IDL マッピング
- コモンビジネスオブジェクト（CBO），AP モデルの提供
  - ・登録されている CBO の検索・情報提供
  - ・登録されている AP モデルの検索・情報提供
- 既存オブジェクト活用環境の提供
  - ・登録済みのビジネスオブジェクトを使用して新規アプリケーションを構築するための機能の提供
  - ・登録済みの AP モデルを使用して新規アプリケーションを構築するための機能の提供

## 3.5 実行環境の提供

上記の開発環境により生成・構築されたアプリケーションオブジェクトおよびビジネスオブジェクトを, 安全・確実かつ効率的に実行させるための実行環境を提供する．

### 3.5.1 ビジネスオブジェクト制御

1） ビジネスオブジェクト呼び出し制御

アプリケーションオブジェクトがビジネスオブジェクトを呼び出すと，実行環

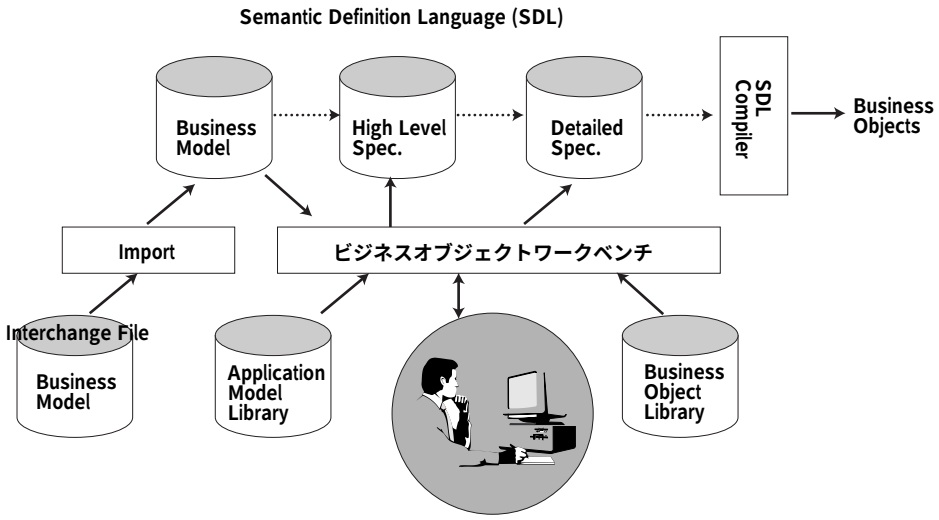


**図 7　BOCF 統合開発環境**

境はそれを仲介し，メタオブジェクト経由でビジネスオブジェクトに制御を渡す．ビジネスオブジェクトは，複数のアプリケーションからの同時並列的な呼び出しに対してそれぞれ独立に全く同様の処理を行わなければならない．

2)　ビジネスオブジェクトの動的構成変更

ビジネスオブジェクトは，動的に構成変更が可能でなければならない．構成変更とは以下のことを指す．

・ビジネスオブジェクトの物理的配置・並列処理数の変更

・ビジネスオブジェクトのメソッド名と呼び出し識別子との対応の変更

ビジネスオブジェクトは，その呼び出し頻度，稼動率，稼動ノード等がモニタリングされる．

3)　セマンティック制御

また，ビジネスオブジェクトをセマンティックに制御できなくてはならない．開発環境において，ビジネスオブジェクトの開発・カスタマイズを行った際に，IDL レベルにて表現できないセマンティックな属性を，SDL にて記述される．そのコンパイルの成果物は，IDL とメタオブジェクトとなる．メタオブジェクトは，ビジネスオブジェクトと対で生成され制御されなければならない．実行時に，ビジネスオブジェクトが呼び出されると，その呼び出しタイミングにて，対応するメタオブジェクトを呼び出し，SDL にて定義された「意味的振る舞い」を実行させる．

### 3.5.2　アプリケーションオブジェクトの制御

開発環境に入力したアプリケーション情報（アプリケーションの振る舞いの定義）は，アプリケーションオブジェクトとして実装される．アプリケーションとは，ある業務処理においてユーザが意識する論理的なひとかたまりの処理であり，一つ以上のビジネスオブジェクトの呼び出しとそれに関連した処理を記述したものである．

アプリケーションの振る舞いとは，以下のことを指す．

・ビジネスオブジェクトの呼び出し
・コミットおよびリカバリの宣言
・例外処理
・異 CORBA/非 CORBA 環境へのアクセス
・システムリソース（DB，ネットワーク，プリンタ等）へのアクセス

### 3.5.3 アプリケーション・サービスの提供

アプリケーションを実行していく上で必要となるサービス機能を提供する．

・トランザクション・サービス
特に，コミット/リカバリの宣言に関連して，トランザクション制御は重要な機能である．ここでいうトランザクションとは，即時性・同期レベルが高いものから低いものまでを含んで扱えなくてはならない．

・例外処理サービス
アプリケーション実行時に発生するシステム例外・ユーザ例外を統一的に制御する仕組みを提供することにより，ユーザの例外に対する考慮を極小化する．

・モニタリングサービス
『アプリケーションの振る舞い』に関連する情報を記録することにより，アプリケーション/ビジネスオブジェクトの稼動状況を監査・証跡可能とする．

### 3.5.4 システムファシリティの提供

アプリケーション制御・ビジネスオブジェクト制御のベースとなる基本機能，およびユーザ支援機能を，システムファシリティとして提供する．メッセージング（トランザクションを含む）機能の提供，CORBA/IIOP 以外の，システム間通信プロトコルのサポートすることによりユーザは異種環境へのメッセージングを，プロトコルを意識することなく行うことが可能となる．例えば

・X/Open DTP
・HTTP
・非 CORBA 分散メッセージング基盤連携
・汎用機接続プロトコル
ex. SNA LU 6.2，OSI-TP 等
・既存システム/ソリューションのゲートウェイ機能の提供
分散オブジェクトの考え方以外で構築された既存システムやソリューションパッケージとの連携を可能とすることにより，システムの付加価値を高める．
・エージェント機能の提供
ネットワーク上に存在するあらゆるシステムリソースを有効に活用して，より効率的なアプリケーションの実行環境を提供するためにエージェント機能を提供する．

## 3.6 開発環境と実行環境

BOCF の開発環境と実行環境は，密接な関係を持つ．ユーザが定義したアプリケーションを，BOCF 開発環境の表現方法に従って表現し，アプリケーションオブジェクトおよびビジネスオブジェクト（およびそれらとペアとなるメタオブジェクト）の形態で出力する．その成果物であるオブジェクト群を BOCF 実行環境が実行制御

することにより，目的のアプリケーションを稼動させることができる．両環境は，統合リポジトリを介して結びついている．統合リポジトリには，下記の内容が格納される．

・アプリケーション・モデル（AP の雛形）
・ビジネス・オブジェクト
・ビジネス・オブジェクト・ソース
・アプリケーション・オブジェクト
・メタ・オブジェクト
・論理呼出し名と実オブジェクト/オペレーション名のマッピング情報

### 3.7 相互運用性と可搬性

1) 相互運用性の確保

相互運用性は以下のことを意味する．

① 本開発環境で作成されたアプリケーションオブジェクト/ビジネスオブジェクトが分散された本実行環境で稼動可能なこと．CORBA 対応プロダクトは分散ノード間で同一である必要はない．

② 本開発環境で作成されたアプリケーションオブジェクト/ビジネスオブジェクトが，他環境上の OMG/BOF 対応オブジェクトを呼び出し可能なこと．

2) 可搬性の確保

可搬性は以下のことを意味する．

① 本開発環境で作成されたアプリケーションオブジェクト/ビジネスオブジェクトが，他の OMG/BOF 対応プロダクト上で稼動可能なこと

② 他の OMG/BOF 対応開発環境上で作成されたアプリケーションオブジェクト/ビジネスオブジェクトが本実行環境上で稼動可能なこと．

これは，バイナリ互換は基本的に不可能なので，開発環境の中間生成物を移殖し，移殖先でメイクを行うことになる．

## 4. BODTF 要求仕様との対比

OMG/BODTF では『OMG BODTF RFP-1』として BOF に対する要求仕様をまとめている．BOCF としては，その要求仕様は基本的にすべて満たすべきであるが，そのうちの重要要件について，BOCF の立場から考察を行なって見る．

1) インターオペラビリティ

業務の分析・設計フェーズの結果を，ビジネスセマンティックを記述可能な言語によって表現することにより，分析・設計の下流工程（開発以降）を，標準化することができる．また，この言語で定義されたビジネスセマンティックをプログラミング言語にマッピングすることで，プログラミングに必要な多くの構成要素を自動的に生成することができる．これにより，プログラマは最小限のプログラミングで，ビジネスオブジェクト，アプリケーションオブジェクトの開発を行うことが可能となる．この一連の流れは，あらゆる環境において共通とすることが可能である．

また，実行フェーズにおいては，BOCF をサポートする異なる環境にまたが

ってアプリケーションオブジェクト/ビジネスオブジェクトを移植および協調動作させることができる．

2） 実装との分離

セマンティック記述言語（SDL）を使用しビジネスセマンティックを記述することは，分散オブジェクト層よりも上位の所謂アプリケーション層で行われる．このため，分析・設計結果を，下位層に依存しないかたちで表現できる．開発においては，プログラミング言語へのマッピングによりプログラムの大部分が生成される．この生成されるソースコードは，開発者の望む分散オブジェクト層やプログラミング言語に対応していなければならないが，このマッピングは BOCF が実施することにより，開発者から，それらの違いを隠蔽することができる．

3） ビジネスオブジェクトの拡張性

ビジネスオブジェクト，コンポーネントの拡張は，そのビジネスオブジェクトの持つビジネスセマンティックの拡張である．分析，設計フェーズでのこれらの要求に対しては，分析・設計者が，ベースとなるビジネスオブジェクトのセマンティックを変更し，新しいメタオブジェクトを作ることで拡張を行う．また，メタオブジェクトの拡張だけでなく，ビジネスオブジェクトの拡張により，属性の拡張を可能とすることができる．

4） 再利用性

ビジネスオブジェクトの再利用性を向上させるために，開発時においてそれらのビジネスセマンティックのブラウジング，カスタマイズ，ユーザアプリケーションへの組み込みのサポートを行わなければならない．

また，BOCF のビジネスオブジェクトを開発するにあたっては，そのセマンティックをセマンティック定義言語で定義しなければならない．ビジネスオブジェクトを提供しようとするベンダーは，BOCF の実装ソフトウェアを使用することが必須であるが，定義されたビジネスセマンティックをセマンティック定義言語を介して表現することにより，再利用性を確保することができる．

5） 開発容易性

アプリケーションおよびビジネスオブジェクトは，SDL によりビジネスセマンティックが記述される．これは，下位層を意識することなく，あらゆる環境において共通である．このビジネスセマンティックをプログラミング言語にマッピングすることで，開発者のソースコード記述量は極小化される．

また，既存ビジネスオブジェクトの表示およびビジネスシステムへのビジネスセマンティックの組み込み機能を開発環境において提供することにより，ビジネスオブジェクトの再利用も促進される．

6） レガシーシステムとの連携

レガシーシステムとの連携は，ビジネスオブジェクトが直接レガシーデータベースにアクセスする場合と，レガシーアプリケーションを呼び出す場合がある．どちらの場合においても，ビジネスオブジェクトからは外部リソースへのアクセスとして一貫した操作性でアクセスできることが望ましい．

これを実現するために，ビジネスオブジェクトとレガシーシステムの間に，レ

ガシーシステムへの操作を集約したラッピングオブジェクトを作成することができる．

7） 非 CORBA との関係

非 CORBA との連携はレガシーシステムとの連携と同様に，一貫した操作性でアクセスできることが望ましい．特に非 CORBA 分散オブジェクト基盤に対してのアクセスは，CORBA 基盤と操作性を一致させるゲートウェイオブジェクトを共通ファシリティとして提供することができる．

8） ビジネスオブジェクトとメタデータの仕様

ビジネスオブジェクトの設計において，IDL を包含する SDL を使用することで，そのセマンティックを実現するためのメタオブジェクトが生成される．実行時において，このメタオブジェクトはビジネスオブジェクトと連係して動作する．

ビジネスセマンティックを表現するために，制約，ルール，ロール，ステート，依存関係，エラー条件などを記述可能な SDL を提供する．

9） コンフィグレーションマネージメント

大きな，分散された，異なる環境においてネットワークシステムのオペレーションを邪魔することなくビジネスオブジェクトやその他のコンポーネントのアップグレードをしなければならない．これは，実行中のオブジェクトの実装の変更や異なる環境にまたがった関係を調和させることを含む．ビジネスオブジェクトは適所に配置された後に構成されなければならない．また，開発時には予想できなかったオブジェクトからもサブクラス化され，利用ができなければならない．オブジェクトは場所や実装によらずに他のオブジェクトと協調動作ができなければならない．この要件は運用要件であり，課題として認識しなければならない．

10） インスタンス特化

インスタンス特化とは，ユニークな能力を持つオブジェクトインスタンスにメソッドや状態を追加する能力である．その拡張は単独の問題を持つソリューションにとっては一時的なものであるかもしれないし，継続的な要求のために永続化されるかもしれない．いずれにしても，再利用性実現の枠組みの中で実行されることが可能となる．

## 5. おわりに

OMG/BOF をベースとしたビジネスオブジェクト制御フレームワークについて述べてきた．BOCF の概要および必要とする機能の概略が整理できたのではないかと考える．

分散オブジェクト分野における CORBA の地位が確立し，ORB 部分の議論が収束の方向にある現在，その上位層である OMG/BOF（BOCA）の領域が，熱い議論の対象となっている．また，OMG の動向とは別に，米国では，コマースネットが eCo フレームワークにより，ビジネスオブジェクトの再利用に対する解を与えようとしている．一方，日本国内の動向としても，ビジネスオブジェクト推進協議会（CBOP）が 1997 年 12 月に設立され，本格的な活動を開始している．ビジネスオブジェクトをめぐる動きは非常にめまぐるしく，かつ活発になってきており，いまや情報処理業界

を席捲する勢いである．

当社においても，上記の BOCF に相当する実装プロダクトの開発を行なっており，1998 年 5 月にも $\beta$ 版を出荷予定である．今後も，この分野の重要性を十分に認識し，業界動向を把握しながらフレームワークを開発していきたいと考える．

最後に，BOCF のコンセプトとプロダクト開発に参加していただいたソフトウェア開発部，情報技術部の諸氏に感謝の意を表する．

---

**参考文献**
［1］ CORBA Component Model RFP.
［2］ CORBA Component Imperative.
［3］ 日経データプロオブジェクト技術の最新基礎知識, 1996 年 5 月.
［4］ 月刊ジャバワールド, 1998 年 4 月.
［5］ 古山一夫, DCOM ガイドブック. オーム社.

**執筆者紹介**

北　川　達　朗（Tatsuro Kitagawa）

1977 年埼玉大学理学部数学科卒業．同年日本ユニシス(株)入社．汎用機の OLTP パッケージの開発・保守，オープン系の基幹システムの技術サポートを経て，分散オブジェクト関連の企画・技術調査を担当．現在，新事業企画開発部市場開発室に所属．

遠　藤　英　幸（Hideyuki Endo）

1987 年福島大学教養学部特別理科物理学専攻課卒業．同年日本ユニシス(株)入社．オブジェクト指向型開発ツールの開発・保守を経て，分散オブジェクト関連の企画・技術開発を担当．現在，新事業企画開発部市場開発室に所属．