OMGのビジネス・オブジェクト標準化と適用上の課題

OMG Business Object Standards and Their Application Issues

岩 田 裕 道

要 約 ビジネス・オブジェクトあるいはコンポーネントの組み合わせによるシステム構築やソリューション・パッケージの活用は，究極のソフトウェア再利用である．ビジネス・オブジェクトは，ビジネス場面に所在する事物をビジネス担当者の視点で捉えたオブジェクトであり，その標準化はこれをさらにオープンな形で進展させる．

本稿では，これらの背景とビジネス・オブジェクト標準化の意義を考察し，OMGが進めている当該標準化活動を概説する．これらの活動はメタモデル，モデリング，仕様化言語，アーキテクチャやサービスなど多様な側面を持つが，特に，ビジネス・オブジェクトの相互運用を可能にする技術基盤である，ビジネス・オブジェクト・ファシリティを中心に考察する．この標準は現在（1998年2月）なお審議中であり確定していない．その見通しや今後の展開を考察するとともに，技術面から見たユーザ企業にとっての今後の課題について言及する．

Abstract The systems building by assembling business objects/components and the use of solution packages are the ultimate software reuse. The business objects are objects, which represent the things residing in the actual business area from the business personnel's viewpoint. Standardization of business object helps it to grow in more open way.

In this article, the background of above-mentioned trend and the meaning of the business object standardization are discussed. And the ongoing standardization activities in OMG are outlined. Though these activities have a various aspects such as metamodel, modeling, specification language, and architecture and service, the main part of this paper is devoted to the Business Object Facility, which is the technical infrastructure enabling the interoperability among deferent business object implementations. Presently (Feb. 1998) this standard is under discussion and not yet finally defined. This paper discusses its expectation and future extension, and also several issues which should be addressed by user enterprises, at the technological point of views.

1. ビジネス・オブジェクト

1994年に発刊されたOliver. Simsの著書[1]によれば，本稿で扱うビジネス・オブジェクトやコンポーネントという概念は，Cooperative Business Objectというアイデアのもとに，1989年に当時IBM UKにいたこの本の著者らによって構想され，1994年にはこれを実現するためのNewiという製品が出荷されている．この製品には，2章以降で紹介する多くの技術要素やアーキテクチャがすでに組み込まれていた．同時期にオブジェクト・マネジメント・グループ（OMG）では，ビジネス・オブジェクトに関するスペシャル・インタレスト・グループ（SIG）が発足し，翌年にはビジネス・アプリケーション・アーキテクチャと称するWhite Paperが出されるとともに，関連するいくつかのタスクフォースが設定され，活動を始めている．

ビジネス・オブジェクトという概念が誕生した正確ないきさつは知らないが，それが現実味を帯びて認識されてきたのはここ数年のことである．OMG におけるこの概念や仕様の標準化作業は，本稿の執筆時点（1998 年 2 月）ではまだ進行中であるが，実際にはこれを先取りして実装した製品が発表されたり（IBM のサンフランシスコ・プロジェクトなど），また ERP パッケージなどの実装にも影響を与えている．

1.1 背景と定義

ビジネス・オブジェクトあるいはコンポーネントの組み立てによるアプリケーション構築という考え方が浮上してきた背景はさまざまあるが，情報技術の観点からは次のように整理できる．

- オープン・システムの発展形態としての多層クライアント/サーバ・コンピューティングの実現手段として，分散オブジェクトという考え方が注目されてきた．
- ビジネスのモデリング技法としてオブジェクト指向技術が普及しつつある．
- Java 言語とそのアプリケーション開発環境が，プラットフォームとの独立性，言語の簡便性，インターネットとの親和性などの特性から急激に注目を集めている．
- OMG などの分散オブジェクトの標準化活動が，プラットフォーム技術からさらにビジネスよりの標準に移行してきている．

最後の項の活動のかなめに位置するのが本稿の中心である共通ビジネス・オブジェクトおよびビジネス・オブジェクト・ファシリティの標準化である．

「ビジネス・オブジェクト（BO）とは，ビジネス領域でアクティブなものの表象であり，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．BO は例えば人，場所，概念を表す．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になる．」[2]言い換えれば，その表現形式は問わないがビジネス領域から見た，つまりビジネス専門家やユーザの観点で捉えたオブジェクトである．通常，オブジェクトという用語はもっと広い範囲を指すが，GUI で対象となるウインドウ，アイコン，メニューなどのオブジェクトや，システム開発の設計段階で情報処理の都合上人工的に作るオブジェクト（テーブルや制御用・管理用のオブジェクトなど）は一般には BO ではない．これらはテクノロジ・オブジェクトと呼ばれ区別される[3]．アプリケーションは，構造的にはこれらの BO とテクノロジ・オブジェクトを使って，ユーザ・コードで接着したものであり，特定のビジネス問題のソリューションを与える．すなわち，アプリケーションは BO のユーザである．例えば「顧客」という BO は，オーダ・エントリ，出荷・請求システム，顧客管理といった多様なアプリケーションで再利用される．

1.2 標準化の意義

一般にビジネス概念やビジネス構造は，企業内のシステム間，同じ業界の企業間，あるいは異なる業界間でかなりの共通部分があることは容易に推測できる．「コンサルタントなどの指摘によれば，特定のビジネス概念や構造の 80% が複数の業界で共通であり，特定の業界のビジネス間では 95% が共通である．」[3]といわれている．これらの共通項が標準化されてコンポーネントとして再利用できたとすれば，（この数

字を単純に引用すれば）残りの 20% あるいは 5% の差分だけをユーザがコード化すればよいことになる．この数字はにわかには信じがたいが，標準の BO やコンポーネントによる組み立て主体のシステム構築（「New World」[1]）が現実のものとなる可能性は高い．ERP パッケージなどのベンダーが，このような標準の BO を再利用可能なコンポーネントとして実装し，それらの組み立て品としてパッケージが構成されれば，顧客は異なるベンダー製品を部品単位で利用・置換できる．分野ごとに特化した質の良い部品の流通も期待できる．

BO はオブジェクトであるから単独では存在しない．他の BO と静的および動的な関連を持つ．したがって BO の標準化はビジネス・パターンの抽出につながると考えられる．オブジェクト指向システム開発では，対象となるビジネス領域のモデリングを最初に行うが，この作業はかなりの技術と訓練を必要とする．標準パターンは，この作業に対して優れた参照モデルを与え，作業の簡素化と品質の向上に貢献する．

ソフトウェア技術の発展はその再利用の歴史と捉えることができる．オペレーティング・システムの登場は，ファイル処理やプリンタ制御などの基本機能をファームウェア化し，コンパイラの登場は行列計算や関数計算をファームウェア化した．次にミドルウェアがファームウェア化された．トランザクション処理やデータベース機能，セキュリティ機能がファームウェア化された．すなわち，ユーザ・コードの一部を基本的なものから高度なものへと規格化して固定し，これらの部分を再利用してきたことになる．BO の標準化はその最後に残った部分のファームウェア化，すなわち究極のソフトウェア再利用である．BO の標準化は，このような流れとオブジェクト指向技術の進展との合流点として，偶然ではなく必然的に出てきたものと理解することができる．

2. 標準化の概要と要件

OMG は，オープンな分散オブジェクト環境を実現するための標準的な概念仕様を定めることを目的とする，国際的な非営利団体である（1989 年に設立され，会員数は 800 社以上）．CORBA に代表されるプラットフォーム・レベルの標準化がほぼ一段落して，現在の活動は，さらにビジネス・レベルの標準化へとその焦点が移行するとともに，標準化の対象範囲が拡大されている．

2.1 ビジネス・オブジェクト・ファシリティの位置づけ

ビジネス・オブジェクト（BO）およびビジネス・オブジェクト・ファシリティ（BOF）の位置づけは図 1 の通りである．最上部にはユーザ・アプリケーションがある．その下層には縦割りの各ビジネス領域に属する BO（ドメイン BO：DBO）とこれらに共通の BO（CBO）が位置する．すなわち BO（DBO と CBO）は，その上位のユーザ・アプリケーションによって再利用される．最下部はソフトウェア・プラットフォームを表し，分散基盤としての CORBA や共通オブジェクト・サービス（COS）群などから成る．BOF はこれらの基盤を使用して，DBO や CBO の相互運用をつかさどるアーキテクチャと高位のサービスを提供する．

OMG の活動の中で特に BO に直接関係する活動は，オブジェクト分析・設計タスクフォース（OA & DTF）といくつかのドメイン・タスクフォースが実質的に進め

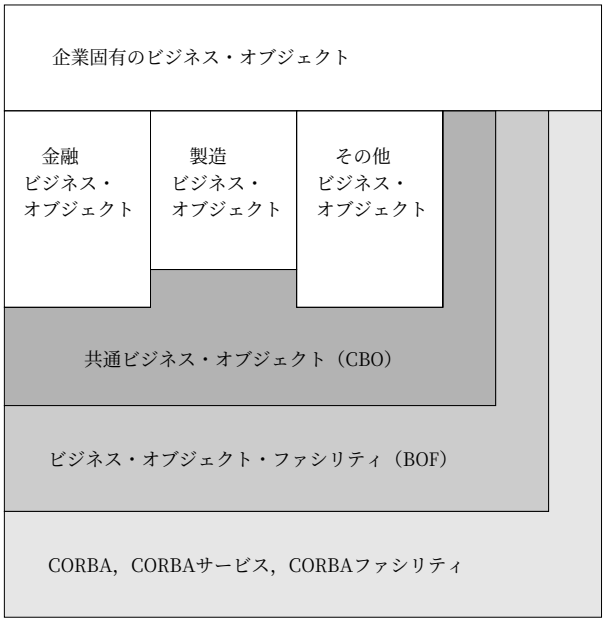

図 1　BOF の位置づけ（OMG[2]より）

ている．OA&DTF の主要な活動としては，統一モデリング言語（UML）[4][5]とメタオブジェクト・ファシリティ標準（MOF）[6][7]の確定がある．UML はオブジェクト・モデリングのための標準図式記法を与える．MOF は，リポジトリのメタモデルを表すと同時に，UML のメタモデルなど他の各種メタモデルを統制する上位のメタモデル（つまりメタメタモデル）を与える．

ドメイン・タスクフォースは，この図の縦割りのビジネス領域に対応するもの，例えば製造，金融，テレコム，EC，医療などの各タスクフォースと，これらに共通する事項を扱うビジネス・オブジェクト・ドメイン・タスクフォース（BODTF）がある．縦割りのドメイン・タスクフォースは，各ビジネス・ドメイン固有の BO の標準化を担当している．例えば製造ドメインでは現在 PDM の標準化[8]などを行っている．

BODTF は CBO と本稿の中心テーマである BOF の標準化，およびワークフローの標準化[9]を担当している．1996 年初頭に CBO と BOF に関する提案要求（略称 CF RFP-4 現在は BODTF RFP-1 ともいう）[2]が発行され，1997 年 1 月にいくつかの具体的な提案仕様が提出された．CBO に関しては二つの提案があり，現在なお審議中である．BOF については，当初互いに競合する 6 件の提案が出され，その後これらの統一に向けて評価ワーキング・グループが設定されたり，公開討論や投票が行われた．昨年後半には競合する三つの提案を含む四つの仕様に集約され，さらにこれら三つの競合する提案が一つの仕様に集約された．したがって現在の BOF は互いに補完しあう二つの仕様から構成されているが，まだ審議中であり確定仕様には至っていない．その内容は 3 章で概説するが，その前にもとの提案要求に上げられている要件を見ておきたい．

2.2 コンポーネントの要件

BOF とは「分散オブジェクト環境の中で，BO が協調的なアプリケーション・コンポーネントとして動作するために必要な機構（アーキテクチャやサービス）」のことである[2]．コンポーネントというのは，設計ならびに実行時の 1 機能単位であり，BO を相互運用・再利用・移植性という視点で捉えたものである．ビジネス領域レベルでのソフトウェア部品と考えてよい．コンポーネントのソースは多様にある．既定のもの；購入ないし再利用されるもの；新たに定義されるもの；低位の OO 言語で作られるもの；他のコンポーネントから組み立てられるもの，などである．

BOF の提案要求では，各ベンダーが提案する仕様に対する要件として，非機能的な必須要件（requirement）と技術基準（criteria）とが掲げられている．前者の要件の主なものは次の通りである．

- 相互運用性：コンポーネントは，以下のような事項とは独立に互いに協調できなければならない――ビジネス・セマンティクスの実装，エンジニアリング技法，ソース言語，OS，各ビジネス・ドメイン．
- 拡張性：BO もコンポーネントも任意のユーザ企業で使えるように拡張できなければならない．そのための手段を備えていること．
- 再利用性：多様なアプリケーションで再利用できなければならない．BO は大きな市場性を持つ必要がある．
- 開発や導入のしやすさ：BO の開発や利用，情報システムへのそれらの配備が，一般のシステム開発者やユーザの観点から容易に行えるものでなければならない．
- セキュリティ：オブジェクトに関する情報の機密が保たれ，セキュリティ・ポリシが励行できるようになっていること．

これらは必須要件（8 項目ある）の一部であるが，これらの必須要件とは別に望ましい要件として，レガシー・アプリケーションとの統合，メタデータのサポート，多言語サポートなどがある．

最近 OMG の日本の代表機関である J-SIG は，単なる言語の違いを超えて，国や地域の法律・制度・商慣習などの違いを十分認識した，BO の国際的な相互運用性が重要である主旨の提起を行い[10]，OMG ではこの概念を regionalization と呼んでいる．これが OMG 内でどのように扱われるかは今後の問題である．

解決されるべき技術課題として 23 項目の技術基準が上げられている．その一部を以下に列挙する．これらの中には，すでに共通オブジェクト・サービスや他のファシリティでプラットフォーム・レベルではカバーされているものもあるが，ビジネス・レベルで扱うにはさらに抽象度の高いインタフェースとサービスが必要である．

- 変更とイベントの通知：BO の属性や関連の値の変更や操作の起動・終了，状態遷移，あるいは特定の事象の発生を，必要な BO に通知する．
- アクティブ・ビュー：BO の一部の側面を取り出して，もとの BO の特化された表現として独立に扱う．企業モデルに存在する BO がビジネスの進化で拡張されても，特定のアプリケーションでは必要なビューのみを見ていればよい．
- 透過的な永続オブジェクト管理：一般にデータベースなどの外部記憶を必要と

する永続オブジェクトについて，異質な DBMS を含むような環境でも，ユーザや開発者が外部記憶の必要性やその異質性を気にしなくてよい．

・入れ子トランザクション：一つのトランザクションの中から別のトランザクションが起動できる．

・構成管理：分散した異質な環境で，全体のシステムの運行を中断せずに，コンポーネントを追加・変更・拡張できる．

3. ビジネス・オブジェクト・ファシリティ提案

既述のように CF RFP-4（BODTF RFP-1）に対する BOF 標準は現在次の二つの仕様に集約されていて，その改良と評価作業が進行中である．

・ビジネス・オブジェクト・コンポーネント・アーキテクチャ（BOCA）

・相互運用フレームワーク（Interoperability Specification）

これら二つの仕様は，互いに補完しあう独立した仕様として構成されているが，これらを総称して Combined Business Object Facility（CBOF）と称している．ここでは，その各提案仕様のうち特徴的な事項についてその一部を紹介する．ただし，個々のシンタクスやセマンティクスを子細に検討し尽くしたわけではなく，またこれらの仕様が必ずしも完全ではない部分もあり，したがって一部に筆者の解釈上の誤解があるかも知れないことをあらかじめ断っておきたい．

3.1 ビジネス・オブジェクト・コンポーネント・アーキテクチャ[11][12]

データ・アクセス・テクノロジ社を始めとする 7 社の共同提案であり，大きく二つの部分すなわち，ビジネス・オブジェクト・コンポーネント・アーキテクチャ（BOCA）とコンポーネント定義言語（CDL）から成っている．BOCA は CBOF 全体を統制するもので，CBOF で用いられる多様な概念を規定するメタレベルのアーキテクチャ，つまりメタモデルである．CDL は，アプリケーションを記述するというより，BO をコンポーネントとして記述するための高位のインタフェース定義言語であり，BOCA が定める豊富な概念を言語構成要素として組み込んでいる．これらの位置づけは図 2 にある．

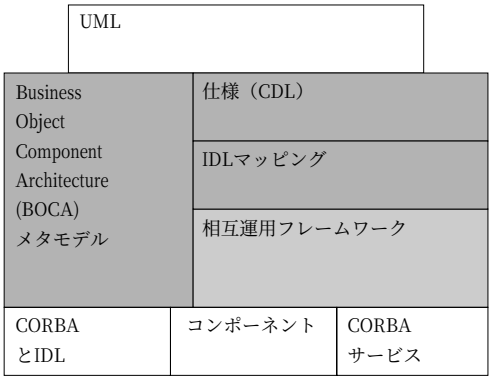

図 2　CBOF の位置づけ（OMG[11]より）

CDL は IDL（CORBA とのオブジェクト・インタフェースを定義する中立言語）のスーパーセットである．BOCA は CDL のメタモデルであり，UML のメタモデルを拡張したものである．従来の OOPL ではメソッドの中に埋め込まざるを得ないような豊富なセマンティクスを，インタフェース仕様記述（クラスの仕様記述に相当）の中で宣言的に書くことができる．この言語の機能上の主な特徴は次の通りである．

- subsystem，entity，process，business event，signal，state set，role といったビジネス・レベルでの概念を，CDL の言語要素として陽に組み込んでいる．
- コンポーネント，属性，関連，操作，イベントなどの構成要素に対して，多様な制約が指定できる．
- 柔軟性と拡張性のための要素が CDL の言語要素として付加されている．
- 派生属性ないし計算属性や条件の記述を，式としてコンポーネント定義の中で陽に記述できる．

BO は subsystem，entity，process，business event の四つに分類されている．subsystem は BO の集まりであり，ビジネス領域の一部を表す．一つあるいは複数のサブシステムが，生産計画とか受注管理といったビジネス・システム・ドメイン（BSD）を構成する．entity はいわゆる実体オブジェクトであるが，振る舞いをカプセル化している．process は操作ないしメソッドの拡張である．違いは，固有の状態（属性や関連）やビジネス規則を持つこと；特定の BO に付随してその中で定義され振る舞うこと；永続的なアイデンティティを持つ一つのオブジェクトであること，である．言い換えれば，process はビジネス・アクティビティを表す．複雑な状態遷移を伴ったり，固有の属性やビジネス規則を持つ複雑な「操作」が process であり，これを操作（operation）つまり単なるアクションと区別している．例えば運輸サブシステムにおける「トラック」というエンティティの定義の中では，「トラック移動」というプロセスがある．このプロセスは目的地という属性や，移動中，休憩中，荷積み中などという複数の状態を持つ．business event はいわゆるトランザクションのことであり，任意のタイミングで一時的に発生する記録さるべきビジネス事象を表す．例えば，注文，出荷，予約などである．

signal はいわゆるイベントを表し，BOCA では操作（operation）のサブタイプとして扱われている．このイベント・モデルは，COS のイベント・サービスをベースにしている．COS のイベント・サービスはパブリッシュ・サブスクライブ（あるいはプロデューサ・コンシューマ）方式の機構を持ち，プッシュ型・プル型双方を扱う．BOCA のイベント・モデルでは，環境内のすべての要素に発生する事象が事実上イベントとして扱える．「signal」文による明示的な指定（explicit　event）以外に，オブジェクトの生成・消去・複写・移動，属性の値の変更，関連インスタンスの追加・変更・削除，集合体のメンバーの追加・除去，プロセスの起動・終了，操作の起動・完了・失敗，状態遷移などの暗黙のイベント（implicit　event）がある．明示的なイベントは，一般に状態遷移規則の中のアクションで起動される．イベント・クライアント（サブスクライバないしコンシューマ）は「trigger」文で特定のイベントへの関心を登録し，実際のイベント発生時にその通知を受ける．通知のための機構は相互運用フレームワークが受け持つ．コンシューマが関心を登録したイベントのみが監視

の対象となる．この概念は，オブジェクトの動的な意味での可視性を高め(対照的に，従来のメッセージ・パシングによるオブジェクト相互作用は，オブジェクト・インタフェースには陽に現れないため，他から可視でない)，かつ実行時の独立性を高める(追加・変更の影響を局所化する)．これは分散アプリケーションの新しいモデル（パブリッシュ・アンド・サブスクライブ・モデル)[13]を支援しようとするものである．

state set はビジネス・レベルでの状態を扱う概念である．例えば「機械」オブジェクトでは，「パワー」という state set はオンまたはオフの状態を持つ．state set は入れ子にでき属性や関連も持てる．「自動車」オブジェクトは「パワー」という state set がオンの状態のときに，「動作」という入れ子の state set は走行中あるいは停止のいずれかである．また走行中の状態では「速度」という属性がある．BO が特定の状態にあることを指示するのに during という概念がある．

role は，提案要求 CF RFP-4 の技術基準にあるアクティブ・ビューに相当する．ベースとなる BO への RoleOf という関連として扱われている．例えば一つのアプリケーション内の「顧客」オブジェクトは，企業モデルに所在する「会社」というベース BO の一つの role であり，この「会社」BO は別のアプリケーションでは「仕入先」という role を持つ．また，同じ「製品」オブジェクトでも，設計段階，製造段階，保守段階の各アプリケーションではその role は異なるであろう．role は，ベースとなる BO の特性のすべてあるいは一部を継承すると同時に，独自の特性を持つことができ，一つの BO が同時に複数の role を持ち，実行時にそれを付加したり除去したりできる．role 概念はまた，従来のオブジェクト・モデルにおいて，オブジェクトを複数の視点で捉えるときに発生する多重継承の爆発を避けることができる．role はサブタイプ関係に似ているが，部分的な継承が可能であること，またそれ自身固有の生涯を持ちしたがって動的であるため，OOPL でのサブクラスとしては一般に実装できない（OOPL では通常，一つのインスタンスが同時に複数のクラスに属したり，その所属クラスを動的に変えることはできない)．role はいわゆる「動的サブクラス」[14]に相当し，「委譲」を用いて実装することになる[15]．関連には RoleOf 以外に，Reference（通常の関連)，Compose（物理的包含)，Aggregate（疎な集約）などの関連種類があらかじめ定義されている．CDL では 2 項関連が前提であり 3 項以上の関連は支援していない（UML は支援している)．

多様な要素に対して多様な制約が指定できる．これらを BOCA ではタイプ・パラメータ，フィーチャ・パラメータという．タイプは BO や集合体，構造を持った一時的なオブジェクト（日付や住所など)，およびデータ型などの総称であり，フィーチャは属性，関連，操作，state set などの総称である．制約の例として，BO やコンポーネントに関しては，isAbstract(そのタイプ自身のインスタンスを持たない)，isLeaf（これ以上特化できない)，isSingleton（インスタンスが一つしかない）などが指定できる．操作では，visibility（Public，Private，Protected などの指定）のほか preCondition（事前条件)，postCondition（事後条件)，isQuery（値を返す操作でかつ副作用を伴わない）などがある．関連と属性双方に共通なものとして，上記の visibility のほか readOnly（他のクライアントが値を設定できない)，isLocked（デフォールト値をサブタイプで変更できない)，target_scope（インスタンスに対するものかタイ

プ自体に対するものか）などがあるが，関連に関しては上記のもの以外に多重度や逆関連の指定，集約関連で親のインスタンスの移動・複写・削除を子のインスタンスにも自動的に伝搬させる指定が可能である．属性と関連には一般化制約（真となるべき論理式）も指定できる．なお，CDL が扱う集合体（collection）には，Set，Bag，List，Ulist，Array，Extent，Home がある．Bag はメンバーの重複を許す順序のない集合であり，List は特定のタイプのメンバーの順序付きのリストであり，Ulist はメンバーの重複を許さない List である．Extent は特定のタイプに属するすべてのインスタンスの集まりであり，Home はそのサブセットである．

これらの静的および動的に作用する制約パラメータは，そのコンポーネントのサブタイプで上書きすることによりカスタマイズすることができる．多様な制約がコンポーネントのインタフェース定義で指定できることは，ユーザ・コードではなく CDL の処理系あるいは基盤のフレームワークが，これらの制約を励行する機能を持つことを意味し，開発者のプログラミング負荷を大きく軽減するのに貢献する．

柔軟性と拡張性をさらに支援するために appliance と apply という機能がある．これは「コンポーネント」を「プラグイン」することのアナロジーで，何らかの要素を appliance として定義して，インタフェース定義の中でそれを apply する．CDL 自身の将来的な拡張用の機能としても働くが，いくつかのものがあらかじめ組み込まれている．例えば，イベントの発生時に評価されるべきビジネス規則の指定とその起動，状態遷移規則の指定，ECARule（イベント―条件―アクション規則）の指定，不変表明（invariant）の指定，シンボルに対する多言語サポート用の label パラメータなどがある．また，アダプターという概念が関連の一種（adapter 関連）として組み込まれている．これは他の BSD にあるオブジェクトを参照するための機構（サロゲート・オブジェクト）である．これは 3.2 節でさらに触れる．

CDL での式表現は，副作用を伴わない式と副作用を許す式に大別され，前者は例えば派生属性（計算属性）の定義で指定でき，いわゆるデータ値（ID を持たないインスタンス）を計算し，返すことができる．後者は主にアクションを伴うビジネス規則で使われ，一般にオブジェクトを返すことができる．インスタンスの集合（エクステント）や集合体に対して，select 文などの問い合わせ式が書ける（ただし機能は限定される）．これらの式の文法は Java 言語の式のサブセットを使っている．

以上は BOCA/CDL の機能面での特徴であるが，かなり意欲的なモデルであり言語である．後述するが，この言語はその基盤である相互運用フレームワークとは密結合しておらず，このフレームワークのための必須要素としては位置づけられていない．この種の 4 GL レベルの言語を標準として規定することの有効性については疑問がある．

3.2 相互運用フレームワーク[16]

これはもともと Electronic Data Systems（EDS）から CF RFP-4（BODTF RFP-1）レスポンスとして別個に出された提案が，CDL 提案と併合され改訂されて CBOF の一部となり，さらにこれと並立していた二つの BOF 提案（IBM 他 1 社の共同提案と System Software Associates の提案）とが CBOF に集約されて一つにまとまったものである．EDS を始め IBM，オラクルなど 9 社の共同提案になっている．

CDL がコンポーネント設計時のファシリティだとすれば，このフレームワークは実行時のファシリティである．すなわち，CDL で記述されたコンポーネント群が異質な分散環境内で効果的に相互運用するための，アーキテクチャとサービスから成る．ただし，このフレームワーク仕様は，必ずしも BOCA と CDL を前提条件にしているわけではない．すなわちコンポーネントの記述言語や実装言語とは独立である．基盤部分には CORBA と共通オブジェクト・サービス（COS）があり，このフレームワークはその上の層を形成する．その包括的なアーキテクチャは図 3（文献[16]にある原図の 1 部を簡略化）にある．この図は，二つのフレームワーク実装（一般に異なるプラットフォーム上にある）に所在する BO の相互運用の様子を示している．図中の「インタフェースとプロトコル」で取り交わされる情報は，ID，ライフサイクル，属性，状態，関連，操作，イベント，問い合わせ，例外条件などである．

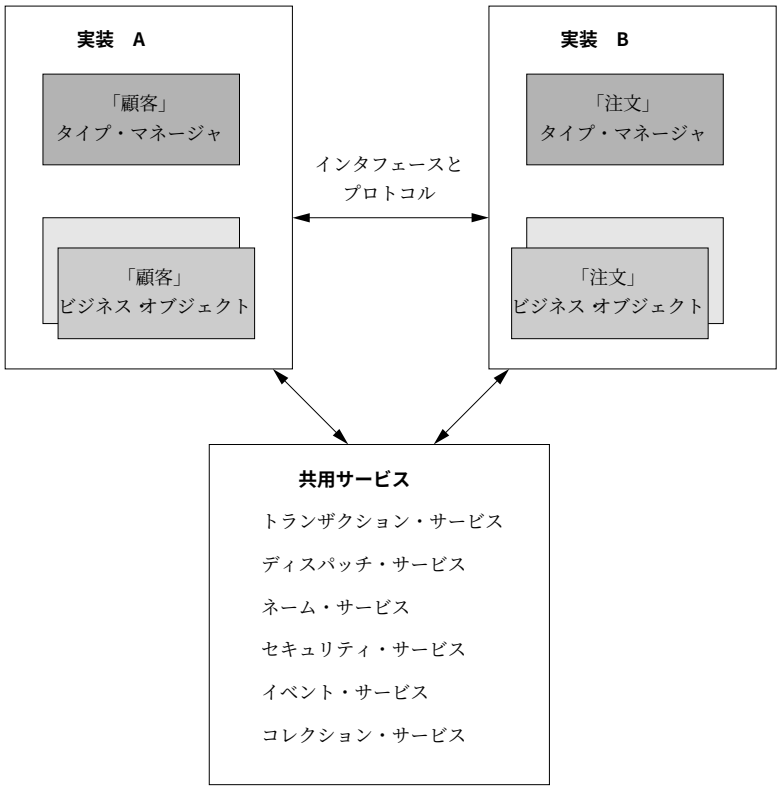

図 3　相互運用フレームワーク（OMG[16]より）

このフレームワークのかなめはタイプ・マネージャ（Type Manager）と共用サービス（Shared Services）である．タイプ・マネージャは，各 BSD 内の個々の BO のタイプごとに存在し，そのタイプおよびサブタイプのすべてのインスタンス（エクステントやサブエクステント）を統制する一つのオブジェクト（テクノロジ・オブジェクト）である．タイプ・マネージャはそのタイプに固有の情報（メタデータなど）とそのエクステントに対して作用する．タイプ・マネージャは，従来の OOPL に見ら

れるオブジェクト・ファクトリ機能やクラス属性，クラス操作といった概念を発展させたもので，分散オブジェクト環境でこれらを動的に扱うためのアーキテクチャである．主な機能は次の通りである．

・ライフサイクルの管理
・問い合わせ
・メタデータへのアクセス
・イベント通知

ライフサイクルの管理は, そのタイプの BO の生成・消去・複写・移動を担当する．インスタンスがインメモリーであることを前提にしており，ID の識別やメモリーにないインスタンスへのオブジェクト参照の解消も行う．永続性サービスはタイプ・マネージャの範囲外である．このフレームワークでは，ID はタイプ名とキーとで構成される（一般に OMG のオブジェクト・モデルではいわゆるユニバーサル ID という概念はない）．

問い合わせ機能は，そのタイプのエクステントやサブエクステントに対する問い合わせを担当し，問い合わせ結果の集合をナビゲートするための反復子（iterator）が提供される．各タイプに定義された属性・関連・操作・引数などのメタデータへのアクセスを行う機能があり，多様なツールで利用できるようにしている．いわゆるリフレクション機能のための受け口になる．イベント通知は，すぐ後で説明する共用サービスの中のイベント・サービスと協調して，そのタイプのインスタンスで発生するイベントを捉え通知する機構である．

共用サービスは，異なる複数のフレームワーク実装が共有するサービス群であり，図 3 の下部にあげられている．BO はトランザクションに参画し，かつ永続オブジェクトであることが前提となる．トランザクション・サービスはディスパッチ・サービスと協調して, 分散環境における複数のフレームワークに所在する BO にまたがって，トランザクションの一貫性や原子性を保証する機構である．COS で規定されたトランザクション・サービスの拡張である．入れ子トランザクションを支援しており，begin メッセージで設定されるトランザクション・コンテキストは，明示的なメッセージ・パラメータとしてではなく内部的にパスされる．ディスパッチ・サービスは，トランザクション内での操作の実行タイミングをきめ細かく制御するための機構である．例えば，副作用のない（更新を伴わない）操作を，コミット処理の前に，影響するすべてのオブジェクトの最終状態を検証するために実行するとか，コミットの後で特定の操作を実行するとか，コミット後一定期間経過後に実行するなどである．ネーム・サービスは COS のサービスの拡張で，セキュリティ・サービスは基本的に COS のサービスに従う．

イベント・サービスは，イベント・クライアント（コンシューマ）による関心イベントの登録とイベント発生時でのイベント通知を行う．イベント通知はタイプ・マネージャと協調して機能する．関心の登録時には，イベント通知のタイミング，イベント発生とその後のアクションとの同期，あるいはリカバリの要・不要などに関するオプションを指定する．同一のイベント発生で複数のコンシューマが存在する場合は，COS のイベント・サービスのイベント・チャネルを通じてブロードキャストされる．

コレクション・サービスは，集合体を扱うサービスであるが，このフレームワークでは仕様が規定されていない．

上記の他に，BO の相互運用を支援する多様な支援要素（機能や構造）がある．前述の反復子は，比較的多量のメンバーを含む集合体に作用する支援オブジェクトである．基本反復子・属性反復子・関連反復子があり，集合体に対して各タプルのブラウズ，選択，タプル内の属性のアクセス，あるいは各種の計算のためのメソッドを持つ．シン・クライアントを支援する一時的オブジェクトとして，アプリケーション・コントロール・オブジェクトとコマンド・オブジェクトがある．アプリケーション・コントロール・オブジェクトは，クライアントと BO との対話セッションの間存在し，特定の会話型タスクの遂行を支援する．コマンド・オブジェクトは，シン・クライアントのリクエストを BO につなぐための瞬間的なアクションを支援する．

複数のビジネス・システム・ドメイン（BSD）間の実行時の疎結合を支援するアダプター機能がある．アダプターは，他の BSD にある BO の代理（サロゲート）であり，メソッド・シグネーチャの変換や ID の対応付けなどを行う．図 4 は，文献[16]にある図をもとに簡略化したものである．

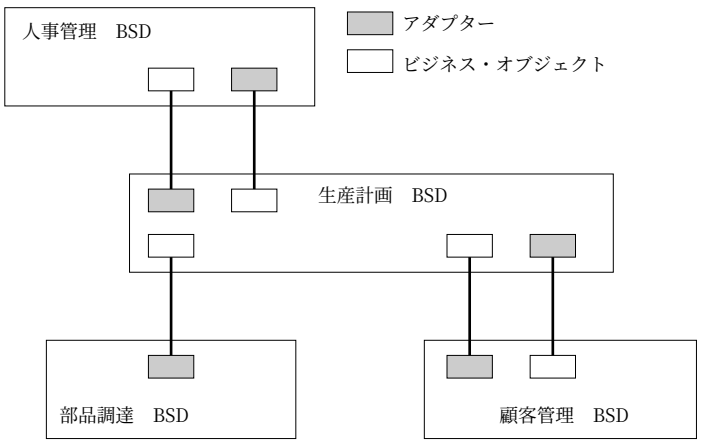

図 4　アダプター（OMG[16]より）

アプリケーションが実行時にユーザ定義の属性を動的に追加したり，別名を付けたり，これらをアクセスするための概念としてプロパティがある．元の BO 仕様とは異なる次元でコンポーネントを扱うことができる．例えば，アーカイブ・サイクルの指定，最後にアーカイブした日付，品質情報，更新履歴などの管理的情報の追加や，顧客名を実行時に得意先名という名前でアクセスするなど．これは各 BO に PropertyContainer オブジェクトを対応させることにより行う．このオブジェクトが上記の動的な属性を保持し，そのアクセスをつかさどる．

支援要素には，他にいくつかの構造体（struct）オブジェクトがある．例えば，BusinessObjectState は，オブジェクトの状態を通知するのに使う構造体で，属性や関連とその値を運ぶ．形式は名前と値のペアからなる NameValue リストである．For-

eignObjectIdentifier は，他の BSD に所在するオブジェクトのアクセスに必要な情報を持つ構造体であり，アダプターが使用する．EventNotice は，イベント発生時にサプライヤとコンシューマ間でイベント通知を行うために交信される構造体である．

OMG 内部での評価グループが指摘しているように，このフレームワークはコンポーネントの相互運用を重視しているが，異なる BOF 実装の上で動く BO のソースコード・レベルおよびバイナリー・レベルでの移植性，および移植可能なパッケージ単位などについて，そのアーキテクチャがよく見えない．CORBA における IDL と他言語とのバインディングに相当するものが必要になる．

4. 標準化の見通しと適用上の課題

OMG のビジネス・オブジェクトの標準化に関係する諸活動はほとんどが現在（1998 年 2 月）進行中であり，これらがいつ頃どのように整合され，一貫した形で公刊されるかは予測できない．複数の標準化作業が互いに技術的に関係しているからである．1. 2 節で述べたように，ビジネス・オブジェクトの標準化は必然の流れであり，その意義が極めて高く，仕様の提案元であるベンダー各社の統一仕様への一致した意志が感じられる．ユーザがビジネス・オブジェクトの標準化を望む限り実現の可能性は高いと考える．以降ではその見通しについての筆者の考えと，今後の展開およびユーザがこれを適用する際に考慮すべき技術面での課題を手短に考察する．

4.1 標準化の見通し

BOF に限れば議論はかなり煮詰まっているので，何らかの形で確定されるのは時間の問題であろう．この技報が発刊される頃には標準の BOF 仕様が部分的にでも確定している可能性は高い．が，その他にも複数の標準が互いに関連していることから，当面それらの関係と内容の整合性が重要になる．主として次の事項である．

・CBO の標準化
・メタモデルの一貫性
・CORBA コンポーネント・モデルとの整合性

CBO については現在二つの仕様があり，これらはまとめて一つの提案文書[17]になっている．一つは IBM 社の提案仕様で，デシマル，時刻，住所といったプリミティブなものと，パーティ（ビジネス・コンタクトの対象となる人や組織）を規定している．他の 1 つは NIIIP の提案である．人が資源を共有し何らかのシステムを使用するという一般的なコンテキストに現れる総称的なオブジェクト，すなわちグループ，ユーザ，役割，ワークプレース，タスク，リソース・アダプターといった CBO とそれらの関連を規定している．しかしこの二つの仕様で必要十分だとは思われない．CBO の標準化には技術的にもいくつかの壁が存在すると考えられる．一つは「共通（common）」という言葉の解釈である．通貨，カレンダー，日付，住所などの具体的でプリミティブなレベルもあれば，場所，人，組織，文書といった抽象度の高いものがある．また顧客，商品，予約といった BO も CBO の対象となる．これらを整理する必要がある．また，各 CBO のセマンティクスを，実際的に意義ある形で定義し同意を計るのもかなりむつかしい．

前述のように，OA&DTF では標準図式言語として UML を確定した．これは工業

製品でいえば設計図面の標準表記法ができたことに相当し，その意義は極めて高い．しかも主要なベンダー（IBM やマイクロソフトも含む）の多くが共同提案していることから，事実上唯一の図式標準になるであろう．同タスクフォースはまた，メタオブジェクト標準である MOF を確定した．CBOF 提案の BOCA（CDL のメタモデル）は，UML のメタモデルや MOF との関係，およびその内容の整合性が今のところ必ずしも明確になっていない．このようなメタレベルでの整合性が要求される．UML，CDL，IDL が扱う諸概念がそれぞれの抽象レベルに応じて一貫した意味を持ち，ユーザから見てシームレスでなければならない．

OMG は 1997 年 6 月に CORBA コンポーネント・モデル RFP[18]を発行した．これは特に JavaBeans コンポーネント・モデルへのマッピングを意識した，CORBA ベースのコンポーネント・モデル仕様の提案要求である．現在，6 件の提案が出て審議中であるが，他の標準と違って実装をかなり重視している点で注目される（Corba-Beans とも呼ばれる）．BOF，UML，MOF などとこのモデルとの整合性が要求される．さらには，ワークフローやリポジトリの標準化，縦割りのドメイン別の BO の標準化とも整合を計る必要がある．

これら複数の標準が，包括的に一貫したアーキテクチャとしてビジネス・オブジェクトを支援するべく形成されるまでには，さらに時間がかかると予想される．加えて，現実のビジネスに適用し普及させるには，実装技術や性能上の課題，これを活用するユーザ企業側の課題もあり，実用経験からのフィードバックが必要である．

4.2 適用上の課題

UML という図式標準の確定により，これを支援しかつ使いこなすためのモデリング方法論が，堰を切ったように出てくるであろう．またコンポーネントの作成やこれらの組み立てによるシステム構築を支援する開発環境を実装した，多様な製品も新たに出てくるであろう．ユーザ企業から見れば選択肢が増えるという便益と同時に，ビジネス・ニーズや企業環境に合った方法論や製品を見極める力をつけることが必要になる．

ビジネス・オブジェクトの標準が確立されてくるにつれ，特定の業種に特化したコンポーネントや統合型パッケージが普及してくると思われるが，ユーザ企業にとっては道具にすぎない．これらの活用は情報システム構築の有力なコスト削減と品質向上手段にはなるが，企業の顧客サービス向上の目的ではない．企業は固有のビジネス・ニーズとビジネス環境を持ち，それが情報技術活用戦略を決める．この戦略は個々の企業固有のものである．既製品を安易に調達して，単にベンダーが指定するプラットフォームで稼働させるだけでは，この戦略目標は達成できない．

企業のビジネス・ニーズを満たすビジネス・モデル（企業レベルおよび業務レベルのオブジェクト・モデル）が明確になっていて，それとこれらの既製コンポーネントのモデルとの違いが正しく識別できねばならない．企業固有の包括的な情報システム構想（情報システム・アーキテクチャ）が明確にされていて，前記の製品群をこのアーキテクチャにどのように整合させられるかが重要である．すなわち，BO の適用を成功させるには，少なくとも次の技術が要求される．

・モデリング：オブジェクト指向技術に基づくビジネス・モデルとアプリケーシ

ョン・モデルの作成．後者は特定のビジネス領域の分析モデルを指す．

・情報システム・アーキテクチャ計画[19][20]：弾力的な企業情報システムを実現するための青写真の計画と設計．特定の製品群とは独立である．アーキテクチャのミスマッチはカスタマイズなどの微調整では修復できない．

・カスタマイズ技術：場当たり的に個別に実施するのではなく，製品の特性とそのアーキテクチャを見定めて，上記のモデルやアーキテクチャとの適合性を検証し，カスタマイズのための作業標準と技術基準を決める．カスタマイズ種類，作業項目，カスタマイズの限界などを決めておく．

・リポジトリ[21]：多様な既製コンポーネントや再利用ライブラリを複数の部門で共有して活用するには，そのための基本的なサービス機能を持ち，ツールの相互運用が可能で拡張性のあるオブジェクト指向リポジトリが必要である．リポジトリは，オープンな分散環境においてシステム構築という業務をシステム化するために不可欠な情報基盤である．

参考文献

［1］ O. Sims “ Business Objects : Delivering Cooperative Objects for Client-Server ” McGraw-Hill, 1994.

［2］ OMG“ Common Facilities RFP-4 : Common Business Objects and Business Object Facility ”OMG TC Document : cf/96-01-04.
（ftp : //ftp.omg.org/pub/docs/cf/96-01-04.pdf で入手可能，以下の OMG 文書も cf/96-01-04 の部分を置換して得られる）.

［3］ R. E. Shelton“ Business Objects White Paper ”（http : //www.openeng.com/busobj.html）.

［4］ OMG“ Object Analysis & Design RFP-1 ”OMG Document : ad/96-05-01.

［5］ OMG“ UML Semantics Ver. 1.1 1, Sep, 1997 ”OMG Document : ad/97-08-04.

［6］ OMG “ Common Facility RFP-5 : Meta-Object Facility ” OMG Document : cf/96-05-02.

［7］ OMG“ Meta Object Facility（MOF）Specification ”OMG Document : ad/97-08-14.

［8］ OMG “ Product Data Management Enablers RFP ” OMG Document : mfg/96-08-01.

［9］ OMG“ Workflow Management Facility RFP ”OMG Document : cf/97-05-06.

［10］ OMG“ CBO RFI Response from Japan SIG ”OMG Document : bom/97-11-21.

［11］ OMG“ Combined Business Object Facility : Business Object Component Architecture（BOCA）Proposal Revision 1.1 ”OMG Document : bom/98-01-07.

［12］ OMG“ CBOF : BOCA Encyclopedia ”OMG Document : bom/98-01-08.

［13］ D. T. Dewire “ Second Generation Client/Server Computing ”（§17.7）, McGraw-Hill, 1997 ;（邦訳：岩田, 他訳「多層型クライアント/サーバ・コンピューティング」, 日刊工業新聞社, 1998）.

［14］ R. Kilov“ Information Modeling : An Object-Oriented Approach ”（§5.1.4 & 5.3.5）, Prentice-Hall, 1996.

［15］ J. Rumbaugh, 他“ Object-Oriented Modeling and Design ”（§4.4.2, 4.4.3）, Prentice-Hall, 1991 ;（邦訳：羽生田栄一監訳「オブジェクト指向方法論 OMT」, トッパン, 1992）.

［16］ OMG“ CBOF : Interoperability Specification ”OMG Document : bom/98-01-10.

［17］ OMG “ Joint Common Business Objects Revised Submission ” OMG Document : bom/98-01-06

［18］ OMG“ CORBA Component Model RFP ”OMG Document : orbos/96-06-12

［19］ Unisys“ Integrated Information Environment : A Conceptual Overview of the Unisys Architecture（Rel. 2.0.0）”Unisys Corp., Mar. 1994.

［20］ 手島, 他「情報システムのパラダイムシフト」（2 章）, オーム社, 1996.

［21］ Unisys“ Universal Repository : Technical Overview（Rel. 1.2）”Unisys Corp., Aug. 1996 ;（http : //www.marketplace.unisys.com/products/urep/manuals/）

執筆者紹介　岩　田　裕　道（Hiromichi Iwata）

1939 年生．1962 年名古屋大学理学部数学科卒業．1962 年 4 月日本ユニシス(株)入社．各種計算受託，OS 開発，言語プロセッサ開発，SE サービス，社内標準整備などに従事．現在、情報技術部技術企画室勤務．

著書・訳書：「ゼロから分かるオブジェクト指向の世界」，（共著，1996 年，日刊工業新聞社），「情報システムのパラダイム・シフト」，（共著，1996 年，オーム社），「多層型クライアント/サーバ・コンピューテイング」，（共訳，1998 年，日刊工業新聞社）