平成 6 年度

ユーザーからみた
オブジェクト指向とは

平成 7 年 3 月

新技術動向研究部会
オブジェクト指向WG

【目　次】

はじめに

（社）日本情報システム・ユーザー協会では、技術研究委員会を平成4年に設置し、その下に新技術動向研究部会を設けた。

そして今年より、マルチメディアWG研究部会、及びオブジェクト指向WG研究部会が新しく設けられた。

システムを考えるとき、それを取り巻く環境の変化や使用するエンドユーザーの高度化がある。オブジェクト指向WG研究部会では、今後のシステムのあり方 について検討を行う上で、最近注目されているオブジェクト指向技術を取り上げ、エンドユーザーの立場からの「オブジェクト指向技術とは」どのようなものか 十数回の検討を重ねてきた。当報告書は、本年の検討結果として中間報告書を作成し、まとめたものである。

1. システムを取り巻く環境激変

経済が低迷する中、円高の定着化を抑え、さらに最近の更なる円高によりどの企業も体質改善を行い、更なるスリム化を行っている。

企業の資産である「人」「物」「金」「情報」の形態も変化していることは既知であろう。

「人」に対するスリム化に関しては、昭和 30 年代から始まり、業務効率化のための省力化、省人化を行っ てきており、最近はホワイトカラーに対する業務効率化の方向に進んでいる。その反面、情報システム部内は、種々の要求が増大し、IS部におけるニーズが拡 大、多様化してきている。

また、長期化している経済低迷の状況で各企業において、情報システム化の狙いが、顧客の確保獲得に重点がおかれるようになり、システム化への要求が変化してきている。

これらは、従来の戦略と情報技術の総合するSISにとどまらず、全社的な業務改革、さらに企業文化すら改革を巻き込んだ形でシステム構築への要求が進んできている。

2. システム環境ボーダレス化

1993 年の米国におけるNII情報構想が発表され、EUをはじめ主要各国で情報スーパーハイウェイ構想に基づき、高度情報通信ネットワークの整備が進められている。

このような状況下でインターネットの世界的な普及、国内のパソコン通信などのブームなど、社会的にネットワーク環境が拡大している。

最近の円高に伴い各企業は、更なるコスト削減を強いられ、アジア地域等のブロック間での生産協力などを行っていることは、既知のことである。

これらに伴い、情報のボーダレス化が進行しているが、システムの接続においても、ビジネスの多様形態に対応するため、情報システム間で水平、垂直方向への接続が要求されるに伴いオープン化が要求されてきている。

ボーダレス化に伴い、システムの運用ルールといったインタフェースのレベルの標準化においても、各国の文化レベルの違いから複雑な調整が要求されてきている。

3. コンピュータ・リテラシーの高度化（WINDOWS）の影響

現在のパソコンの普及にみる、個人レベルのコンピュータ・リテラシーの高度化による、システムへの要求、使用目的の変化がある。

パソコンの性能向上、低価格化は著しく、コンピュータが家電製品化してきている。大量に出回ったコン ピュータに大量にソフトが走る。ソフトも大量に使用され、性能、機能、多様性が向上し、低価格になってきている。ユーザーの使用も趣味からオフィスに、年 齢も高齢へと変化してきており、エンドユーザーは自らPC上の WINDOWS を操作し、ワープロソフト、表計算ソフトなどを使用し、自らの考えで業務を 行っている。

WINDOWS の操作、画面設計、インタフェースを知らずに WINDOWS の標準化に慣れていれば、 WINDOWS 標準に合ったソフトの場合は、マニュアルなしでも初歩的な操作は誰でもできるようになってきている。このようにPCソフトのフレンドリー性 は、基幹システムを構築する上でも多くの影響を与えるであろう。

4. なぜオブジェクト指向なのか

5、6年前からオブジェクト指向は注目されてきており、一部の開発に適用がされてきているが、普及に 至っていない。今まで、開発側の発想としてアプリケーション開発時の開発手法として注目されてきており、開発スピード、開発コスト削減、再利用性等がク ローズアップされてきている。

しかし、オブジェクト指向技術にそった WINDOWS のインタフェースがユーザー操作でのデファクトス タンダードになっている現在、WINDOWS のインタフェースにそったシステム構築が求められている。WINDOWS システムがオブジェクト指向技術を ベースに設計されているため、今後システム設計を行う上で知る必要があるであろう。また、マルチメディアの普及に伴い扱うデータの複雑化や、システムに オープン化が要求され、複雑化するシステムの分析を行う上でもオブジェクト指向技術を知る必要があるであろう。

5. WINDOWS による新たな時代の流れ

エンドユーザーにおいても、ワープロソフトや表計算ソフトを組み合わせ、業務をこなしている。以前は、機能的にも制限があったが、最近の各ソフトはかなりの機能をサポートしてきている。

例えば、マイクロソフトなどでは、WORD、EXCEL、ACCESS などのデータを簡単に連携し、業 務を遂行できる。従来のシステムにおいて、ある業務のシステムを構築する時、機能単位にモジュール化（プログラム化）を行い、ジョブとして各モジュールを 組み合わせていくことであった。

現在のパソコンでの使用形態は、このシステム構築と同様である。例えば、ACCESS に貯めていたデー タを抽出し、EXCEL に取り込み、データの加工を行い、グラフを作成し、WORD で作成された文書にはりつける。このような使用はパワーユーザーでなく とも、簡単に作成できてしまう。従来のシステムでの、抽出、加工、編集のモジュール（プログラム）が、高度に汎用化された部品としてみれる。いわゆる ACCESS が抽出プログラム、EXCEL が加工プログラムであり、従来のタスク、またはアクティビティであり、操作がジョブであろう。

システム化ニーズも多様化し、開発期間の短縮（経費の削減目的以外に即応性が要求されている）が要求される時代であり、業務に合わせて作る考えから、発生都度、目的に合わせて機能を利用するようになってきており、システムの方向性は、このような方向に向 かっているのではないだろうか。

ソフトウエア技術者においても、テクニックで評価する時代から、ツールを知っている量で評価される時代に移行してきたといえる。

現在の基幹システムにまで、このような考え方を拡大していくと、現在各社単位で独自のシステム構築を 行っているが、システムは一企業で閉じてはおらず、他システムと連携して（ネットワーク等で）おり、パソコン同様のOLE機能の連携ができれば、大幅なコ スト削減ができるであろう。

このように、オブジェクト指向技術をベースに今後のシステムのあり方について、ユーザーとして業界に提案をしていくための検討を行った。

本年度は、各メンバーの基礎知識として、オブジェクト指向とは何かを検討を行ってきた。また、昨年末、英国のオリバー・スミス著の「Business. Objects」が発刊され、システム構築での新たなアプローチとして、またオブジェクト指向技術利用の将来像としての新たな考え方が提案された。本検討部会でも「ビジネス・オブジェクト」を取り上げ検討（勉強）し、本報告書にまとめた。

オブジェクト指向WG研究部会活動経緯

【活動経緯】

第1回　94 年8月 23 日（火） JUAS研修室

・自己紹介・主査／副主査の選任
・研究テーマに関する検討
・運営方法の決定

第2回　94 年9月 22 日（木） 証券会館第5会議室

・年間スケジュールの確認
・次回合宿の運営方法の決定
・研究の進め方の討議

第3回　94 年 10 月 21 日（金）～22 日（土） （財）海職業訓練協力センター

・講演会視聴

第1部　分散オブジェクト指向の標準化
第2部　オブジェクト指向概念と方法論の概要

・シグマ会のアンケートの参加について
・研究の進め方について

第4回　94 年 11 月 22 日（火） JUAS研修室

・前回課題事項について
・オブジェクト指向疑問点に関する討論
・配布資料についての説明
・課題事項の提出について

第5回　94 年 12 月 14 日（水） JUAS研修室

・報告書の章立てと担当者の決定
・オブジェクト指向の普及を妨害する要因についての報告

第6回　95 年1月 17 日（火） JUAS研修室

・報告書の進め方
・報告書(案)のすり合わせと検討

第7回　95 年2月 14 日(火) JUAS談話室

・報告書原稿の各内容についてレジメ検討
・今後スケジュール

第8回　95 年3月 14 日(火) JUAS研修室

・各章ごとの内容発表

第9回　95 年3月 23 日(木) JUAS研修室

・原稿の内容についての読み合わせ

第 10 回　95 年4月 13 日(木) JUAS研修室

・原稿の内容についての読み合わせ
・各章ごとの調整

第11回　95 年4月 26 日(水) JUAS研修室

・最終原稿の読み合わせ

第1章　ユーザーニーズ

1．1　経営から見た情報システムへの期待

1991 年から始まったいわゆる「バブル経済の崩壊」という景気後退は、順調な右肩上がりで成長してきたわが国の経済に極めて深刻な影響を及ぼしている。長引く不況、そして急速な円高に対し政府はさまざまな規制緩和をはじめ、公定歩合引下げや所得税減税、 消費税アップ等の税制改革により景気回復刺激策を行っているが、雇用調整・所得減少による個人消費減退の影響は大きく、景気の回復は遅れている状況である。現在、多くの企業で量から質への転換策が進められている中で、情報システムに対する期待はどのように変化しているのだろうか。

（1）真の戦略情報システム（SIS）実現の要求

このような厳しい経済環境の中、銀行の証券子会社設立／生保商品の兼営等、産業界全般の傾向として自由 化、業際化が進行し、また金融系／産業系カード関連ビジネス拡大に伴い販売チャネルの複雑化・商品の多様化が顕著になってきている。事業分野が拡大し複雑化するに従い、各企業は今まで以上に限られた経営資源を高収益の確保できる得意分野に投入する経営戦略を推進する。同時に情報システム化の狙いもバブル経 済時代の基幹業務における大量事務の省力化から顧客の確保・獲得に重点をおいた競争優位を目指せるシステムへと変化しつつある。また、多くの企業がリスト ラクチャリング及びリエンジニアリング（BPR）に取り組んでおり、階層型組織の見直しや社内業務改善を進めるためのシステム化を検討している。このよう に変革しつつある企業経営を支える基盤として情報システムに課せられた役割は極めて重要であり、従来の戦略と情報技術の統合だけのSISではない、全社的 な業務改革を推進する企業文化を前提とした真の戦略情報システム構築が要求されている。

（2）グローバル化、ボーダレス化への対応

1993 年、米国におけるNII構想が発表され、EUをはじめ主要各国で情報スーパーハイウェイ構想に 基づき高度情報・通信ネットワークの整備が進められている。近年、日米間をはじめ国と国や北米／EU／東欧／中南米／アジア地域等、ブロック間の国際経済 活動の活発化に伴い、企業同士のさまざまな経済取引が国境を越え地球規模で展開される状況下、国内の情報ネットワークで企業活動が満足できるはずがなく、 インターネット等のグローバル・ネットワークの普及に伴い、海外情報ネットワークとの接続機能を有する情報システム構築が要求されている。一方、国内でも コンビニにおけるチケット予約・各種支払い受付サービスやガソリンスタンドのICカードによる会員サービス等、ビジネス形態の多様化が進み、異なった情報 システム間で水平・垂直方向の接続が必要とされてきている。このようなネットワーク間及び情報システム間接続を前提にオープンシステム化が推進され、マル チメディ

アやダウンサイジングを含め情報の伝達・表現方法・システムの運用ルールといったインタフェース・レベルの標準化が各国際機関を中心に検討されて いる。

特に、1985 年、米国防総省における「ペーパーレス運動」から発展したといわれる CALS (Continuous Acquisition and Life-Cycle Support)は、EDIで取り扱う取引データのみならず設計図やマニュアル等、すべての企業情報のディジタル化を目指し、企業間／国際間における標準 的な情報交換手順として注目されており、わが国でも具体化に向け官民一体となって検討が進められている。

(3)システム化対象業務の変化

これらの経済環境及び経営環境の変革に伴い、情報システム化対象業務も確実に変化している。従来は事務 省力化を目指し日常業務における大量データを効率的に処理する定型システム(基幹業務系システム)が代表的システム化対象業務であったが、ここ数年、新商品開発や顧客に対する営業支援等、顧客サービスの向上を目的とした非定型情報系システムや資産・収益・コスト・リスク等、企業全体の戦略情報を提供する経 営管理システムの構築が進められている。このような変化に伴い大型コンピュータ／大容量DASD／放射状ネットワーク／業種用端末による大規模オンライン 処理から顧客基本情報を提供し営業店等エンドユーザーに必要な加工を任せるクライアント／サーバーシステムへと、システム運用形態も大きく変革しようとし ている。パソコン通信の飛躍的な普及に加え、マルチメディアの実用化やインターネットを利用したWWW(WorldWide Web)等、さまざまな情報化技術の急速な進展につれ、よりエンドユーザーに近い場所で「Everywhere Computing(どこでも誰でもコンピュータに接する」が大きく広がると予想される(表1－1－1参照)。

表1－1－1

表１－１－１　90年代の情報システム

	これまで（1980年代まで）	これから（1990年代以降）
目的	合理化（省力化）	＋顧客満足（競争優位）
適用業務	基幹系／定型	＋情報系／非定型
処理方式	集中	＋分散
機種	汎用大型機／オフコン	パソコン／ワークステーション
システム形態	ホスト／センター	クライアント／サーバー
プラットホーム	プロプライエタリシステム（独自仕様）	オープンシステム
情報媒体（メディア）	個別メディア	マルチメディア
情報空間	バックオフィス	＋フロントオフィス
	スタンドアローン	＋ネットワーク
	組織内	＋組織間
主体	情報システム部門	＋エンドユーザー部門
ベンダー	メーカー	＋システムインテグレー（SI）
投資	拡大指向	重点指向

（注）＋の付してある項目は、左の項目に加えてという意味
（日本情報処理開発協会「情報化白書　1994」より

１．２ システム部門への期待

システム対象業務の変化により、企業の情報システム構築の主管部門であるシステム部の役割も大きく変わろうとしている。

（1）情報化推進の先導役

30 年以上前、企業にコンピュータ導入が開始されて以来、情報システム部門は社内における唯一の情報シ ステム化担当部門として事業運営に大きく貢献している。特に基幹業務のオンライン化に際しては、ユーザーの要件に基づきシステム開発から維持運用に至るまで広範囲な役割を果たしている。しかしながら最近の情報システム化対象業務の変革や急速に進展する情報化技術動向は、今まで社内情報システム開発／大型コ ンピュータの管理運営／エンドユーザーへの情報サービス提供に専念してきた情報部門の役割を、BPRを前提とした企業全体の情報化推進へと変えようとして いる。

（2）情報システム構築・運用機能の移管

情報システム化対象業務が営業支援や新商品開発へ広がり、ダウンサイジングやオープンシステム化等、運 用形態も複雑化／高度化するに従い、システム開発／維持運用に必要とされる業務知識／情報化技術を有する人材の育成／確保が困難になる。一方、企業競争の 観点から、より短い開発期間でエンドユーザーの使いやすいシステムの実現が求められており、システム・インテグレーションによりメーカーやソフトウエアハ ウスにシステム構築を一括受注したりアウトソーシングで運用管理を全面委託へ移管するケースが増えてきている。

（3）情報システム部門の役割

このように経営者やユーザー部門からの使いやすいシステムを迅速に実現化してほしいという要望を受け て、情報システム部門では開発スピード／コスト／品質を念頭に共通モジュール／部品／パッケージの検討が進められている。すなわち情報システム構築の方法 も、従来の自社開発（MAKE）から基幹データベース維持管理を除くエンドユーザーコンピューティング支援（HELP）へ、あるいは既存ソフトウエア・ パッケージ／部品を購入（BUY）し活用する方向へと変化している。今後、情報システム部門は基幹システムの構築／維持／運用に加えて、最新の情報化技術 を持った専門集団として各ユーザー部門の情報システム構築のコンサルテーションや情報の表現方法／通信方法を含め、システム基盤（インフラストラク チャー）構築を推進していくと同時に、ユーザー部門に設置されているパソコン等のハードウエア／ソフトウエアのレベル管理を含め全社情報インフラの維持管 理を受け持つことになる。

1．3 ユーザー部門の意識改革

情報システム部門の役割が変革すると同時に、情報システムを利用してきたユーザーの立場も変わりつつある。EUC／CSSが急速に発展し情報システム自体が営業／管理等第一線に近づくにつれ、営業部門では顧客サービス向上に役立つ情報システムの活用を考え、また自社の経営戦略立案／決定／評価を行う管理部門では情報システム部門から提供された基本情報を自部門のEUCで迅速に加工／分析する必要が出てき ている。今後、ユーザー部門の役割は自部門の業務改善のため業務知識に加えBPRやEUCを中心とした情報化技術を習得し情報システム部門のコンサルテー ションを得て部門のシステム化推進へと変わっていく。

第2章　情報システム化の動向

本章では、最近の情報システム化の動向の中から、「開発スピードの重視」「WINDOWSの普及」「マルチメディア」「クライアント／サーバーシステム の普及」の4つのトピックスを選び、それぞれがオブジェクト指向技術もしくはオブジェクト指向の考え方に基づいて成り立っていることを示そうと思う。

2．1　機能重視から開発スピード重視へ

従来、システムを開発するにあたって重要とされていたものは機能の豊富さであったが、現在は開発スピー ドの短縮化が最大の関心事になってきた。しかしそれは決して機能を軽視しているのではなく、やむにやまれずにそうなったと見るべきだろう。そこには、従来 のシステム開発における反省がある。すなわち、従来の豊富な機能を盛り込んだシステム開発ではその開発期間があまりにも長いため、システム開発が完了した ときにはすでに使いものにならなかったという反省である。言い換えるなら、ビジネス要件の変化のスピードがシステム開発に要する期間を追い越すようになってきたということである。

また、開発期間を短縮するということは、コストを削減することとイコールであるため今後ますますそのニーズは高まっていくことだろう。

ここでは、開発期間の短縮化に対してオブジェクト指向技術がどのように役立っているかについて見ていこうと思う。

（1）作らないアプローチ

開発期間を短縮する一番良い方法は、できるかぎり開発しないということである。それには以下の方法が考えられる。

①開発する機能をできるかぎり絞り込む

新たに説明するまでもなく機能を絞れば開発期間は短縮する。

②既存システムの機能を最大限に利用する

既存のシステムを1つの部品として捉え、それを再利用することにより新規開発の量を減らす。この考え方はオブジェクト指向における部品化や再利用の考え方と一致する。

③すでに市場に流通しているソフトで同様の機能を実現しているものを流用する

市販アプリケーションを部品として捉え、それを適宜組み合わせることによって、目的の機能を実現する。前項と同様にオブジェクト指向における部品化や再利用の考え方と一致する。

従来、市販アプリケーションは単独で利用されることが多く、他のソフトと連携させることがほとんど不可 能であった。しかし、最近はスプレッドシートにWS上のRDBへのアクセス機能を付加するツールであるとか、パソコンのDBソフトとWS上のRDBを連携 するルーツが充実し、それらを組み合わせることによって新たにシステムを構築することなしに、かなりの機能が利用可能となってきた。また、マイクロソフト 社が開発したOLE2.0 を用いると別々に開発されたアプリケーションソフト同士が実行時にデータを共有することも可能となってきている。OLE技術は OMGのCORBAと並んで、来るべき分散オブジェクトコンピューティングの先駆けとなる技術といえるだろう。

(2)プロトタイプツールを用いるアプローチ

とはいっても、実際の業務アプリケーションをすべて市販アプリケーションソフトの組み合わせによって実現するには課題も多い。例えば、OSのレベルアップへの対応時期が各アプリケーションによって異なったり、異なるベンダーが開発した複数のアプリケーショ ンソフトにまたがる問題が発生した場合に適切な助言をしてくれるシステムインテグレータの不足などが挙げられる。したがって現実的には、情報提供型／意思 決定支援型のシステムを市販アプリケーションソフトの組み合わせで実現し、ミッションクリティカルといわれるような企業にとっての生命線を担うシステムに ついては自社開発を行う、というのが現在のところ一般的といえるのではないだろうか。

自社開発が避けられないとすると短期間にユーザーの要求を把握し、それをいかに作り込むかという点が改めてクローズアップされるが、この問題に対する解答がプロトタイピングツールと呼ばれているGUIを用いたビジュアルプログラミングツールである。これら のツールはほとんどの場合、オブジェクト指向によって開発されており、システム開発者はエンドユーザーに直にGUIを示しながらその場でユーザーの要求を 反映することが可能となってきている。オブジェクト指向技術が有効に活用されている例といえるだろう。

このようにオブジェクト指向の考え方は、開発期間の短縮という課題に対して重要な役割を果たしている。

図2－1－1は、Unix Business Association 発行の Open Directory 1994 年版から抜粋したビジュアルプログラミングツールの市場動向である。若干古いデータであるが、参考にして頂ければと思う。

図2－1－1

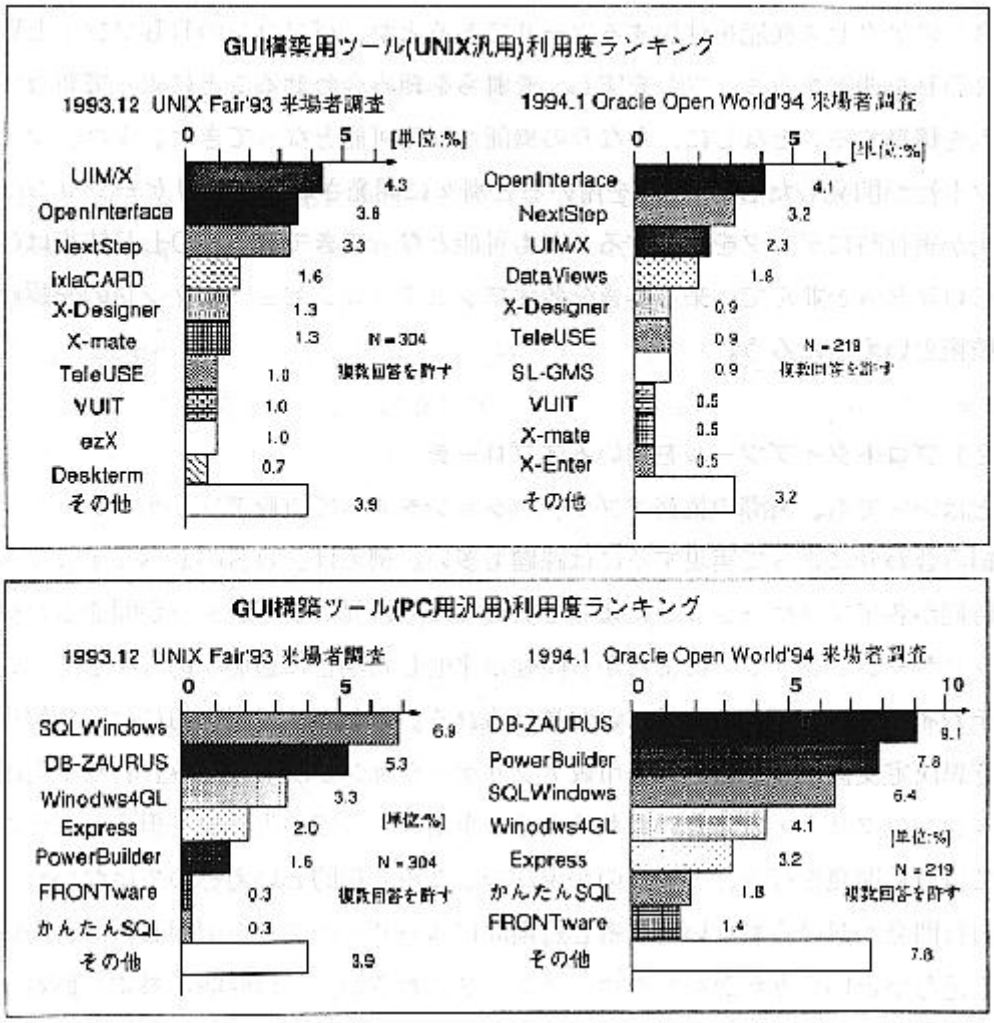


図2－1－1

2．2「WINDOWS の普及」

情報システム化の動向で、特徴的なものの1つに WINDOWS の普及が挙げられる。ここでは、WINDOWS の普及とオブジェクト指向技術の関係を取り上げる。

長らくパソコンOSの主役の座にあったDOSに変わって、現在 WINDOWS は、パソコンのディファク トスタンダードになっている。1993 年5月、マイクロソフト社は、「WINDOWS 3.1J」を発売。出荷ペースを上回る好調ぶりであった。そして、WINDOWS 自体の機能をいかすアプリケーションソフトも数多く出回っている。企業 ユーザーの間で WINDOWS の大量導入もされている。94 年には、WINDOWS-NT が発売され、95 年秋には、WINDOWS 95 が発売される予定になっている。

これほどまでに WINDOWS が急速に普及する秘密は、何といってもDOSとは比較にならないほど操作 性に優れたユーザーインタフェースであろう。WINDOWS は、テキスト画面を

使っていたDOSとは違い、グラフィカルな表示を使用している。ソフトを起動するためにコマンドを必要とせず、アイコンをマウスで指し示すだけでソフトを起動できる。さらに、画面上の好きな位置にウィンドウを開いて必要な情報を 表示できる。このような操作性を実現するためには、「GUI」、「操作系の統一」、「マルチウィンドウ」の3つが必要となる。また、これらの技術を実現可 能にしている背後には、オブジェクト指向技術との関わりがある。では、これからどのようにオブジェクト指向技術が上記3つの技術に反映されているかを見て みよう。

(1)「GUI」

GUIを使う最大のメリットは、ユーザーインタフェースがビジュアルでわかりやすくなることである。例 えば、マウスでドラッグ・アンド・ドロップすることでファイルの複写や移動ができたり、ファイル・マネージャからプログラム・マネージャにドラッグ・アン ド・ドロップすると、プログラム・マネージャにアイコンが登録される等。つまり、ファイルやディレクトリ、周辺機器などをグラフィカルな絵で表現し、各資 源に対する操作はアイコンを通して実現している。

例えば、削除機能をゴミ箱のアイコンに関連付けをしたとき、削除したいオブジェクトをマウスでドラッ グ・アンド・ドロップし、ゴミ箱に入れることにより削除される。現実の動作と、コンピュータ上の動作がGUIによって、違和感なく表現されている。これ は、コンピュータ内の論理的なものを現実世界のものとして実現し、理解しやすくしている。

(2)「操作系の統一」

さらに、WINDOWS の操作性を向上させているのが操作体系の統一である。ソフトの起動と終了、ファ イルの開閉、ウィンドウの操作など基本的な操作を統一しているので、DOSのようにアプリケーション・ソフトごとにばらばらな操作方法をいちいち覚える必 要がなくなった。このことは、一連の操作をオブジェクトとしてモデル化し、各アプリケーションで再利用することによって統一が可能となる。

(3)「マルチウィンドウ」

WINDOWS はその名の通り、複数のウィンドウである。例えば、ワープロのウィンドウと表計算のウィ ンドウを一つの画面上に開ける。異なる情報を一度に見られるので作業の効率が上がる。つまり、1つのディスプレイ上に複数のウィンドウを開く場合、各ウィ ンドウは、別々の作業が行え、ユーザーは、ウィンドウ単位でシステムのやりとりが可能となる。すなわち、性質の異なる多くの作業が1つのディスプレイ上で 行われているが、各ウィンドウは自分自身の持つ情報を他のウィンドウに対して隠蔽していなければならない。また、各ウィンドウはそうした独立性を保つこと の他に、ウィンドウとして、閉じる、動かす、アイコン化する等ができるといった共通する性質を持つ。これらの性質は、オブジェクト指向のクラス概念と一致 している。

WINDOWS の普及した理由は、一言で言えば、ユーザーにとって使いやすいシステムだからである。そ して、その背景にはオブジェクト指向技術がある。現在、WINDOWS だけでなく、OS／2のプレゼンテーション・マネージャ、UNIX上のMOTIFなど様々なところで、オブジェクト指向型のユーザーインタフェースが取り入れられて、広く使われている。また、これらのOS上で動くアプリケーションのプロ グラミングは、マウスのクリックや、メニュー・コマンドの指定など、様々なメッセージが送られてそれを随時処理するといったメッセージ駆動の方式をとって おり、これは従来のシーケンシャル処理を基本とするプログラミング言語とは随分異なり、オブジェクト指向言語のパラダイムに近い。

（4）WINDOWS のオブジェクト指向化の中心技術になるOLE

WINDOWS 3.1 からは、OLEという連携機能を追加した。これは、複合ドキュメントを作るためのものであり、これは今後の WINDOWS にとって非常に大事な技術 になる。OLEで同じようにリンクを張る場合、データを受け取る側（クライアント）は送る側（サーバー）のデータ形式について知る必要がなく、ビットマッ プや、テキストのような汎用のものならOLEのDLLが表示を担当し、そうでない場合はサーバーが表示を請け負う。サーバー側のデータ形式が全く新しいも のでも、OLEは大丈夫である。データとそれを処理するプログラムが結び付いている点が、オブジェクト指向の考えに似ている。

OLE2.0 では、複合ドキュメントの機能をさらに強め、in-Place activation と呼ぶ機能を付け加えた。これを使うと、ダブルクリックでデータを編集するときにサーバー・アプリケーションが、クライアント・アプ リケーションの一機種であるかのように、同じウィンドウで立ち上がる。しかし、OLE2.0 はネットワークに対応していない。この点ではDDEが進んでい る。OLE2.0 の次バージョンは、ネットワークに対応する見込みである。Microsoft はOLEをマルチプラットフォームに提供していく計画であり、将来はマルチプラットフォームでのデータ連携が可能になるとみられそうである。また、ウィンドウズ・アプリケーションをカスタマイズする方法を探している情報システム部門にとって、OLE2.0 を習得することは力強い味方になることだろう。

Chicago では、OLE2.0 を取り入れる。OLE2.0 はOLE1.0 に比べ、オブジェクト指向をかなり強めた。すでに 94 年1月に出荷が始まった Microsoft の EXCEL はOLE2.0 の機能を利用しているが、OSに標準添付するのは WINDOWS NT 3.5 に続き、Chicago が2番目になる。さらに、Cairo では、OLE3.0 を標準装備する。OLE2.0 は、今後 Microsoft が WINDOWS をオブジェクト指向OSへと進化させていくカギを握る技術である。

（5）独自アプリケーションの開発

WINDOWS は MS-DOS と異なりAPIを関数の形で提供する。WINDOWS3.1 アプリケーションのプログラミングはマウスのクリックやメニュー・コマンドの指定など、さまざまなメッセージが送られ、それを随時処理するというメッセー ジ駆動の方式を探っている。これは従来の MS-DOS プログラミングの手法とはだいぶ異なり、C言語のようなシーケンシャル処理を基本

とするプログラミン グ言語にはあまりなじまないところ もある。オブジェクト指向言語の方がパラダイムが近い。1993 年に Microsoft は VisualC＋＋の出荷 を始めた。これは MS-DOS に戻ることなく WINDOWS の中だけで開発ができる統合環境を備えている。さらに、WINDOWS アプリケーション開発に 特化したクラス・ライブラリMFC2.0（Microsoft foundation classlibrary)を備える。メニューやダイアログ・ボックスをビジュアルに作れるだけでなく、ツールバーを作ったり、印刷のプレビューを作った り、OLE1.0 のクライアント機能を簡単にプログラミングできるなど、アプリケーションを作るのに便利な機能を数多く備えている。特にOLE対応のアプ リケーション開発に適している。米国では 93 年末にOLE2.0 用のクラスを備えた VisualC＋＋1.5 の出荷も始まっている。

よりユーザーレベルに近い開発ツールとしては Microsoft の Visual Basic が重要である。日本版はまだバージョンが 2.0 だが、米国ではデータベースサーバーにアクセスするためのODBC（open database connectivity）の部品やOLE2.0 クライアントの部品を持つ Visual Basic3.0 が最新版である。Microsoft は今後、Visual Basic をアプリケーションの共通マクロ言語としても採用していく意向であり、第一弾として 94 年1月に米国で出荷を始めた Excel5.0 は Visual Basic for Application（VBA）を備えている。VBAは Visual Basic の基本機能に加えプログラム中で市販アプリケーション・パッケージの機能を利用できるようにしたものだ。Visual Basic の特徴の1つはウィンドウに配置するための部品を Visual Basic カスタム・コントロール（通称VBX）という形でどんどん拡張できることだ。米国では 100 種を超えるVBXが販売されており、表計算ソフトを 実現するもの、データベースを簡単に構築できるようにするもの、通信プロトコルを提供するものなど、Visual Basic を支える大きな力になっている。Microsoft は 94 年3月にVBXの後継となるOLEカスタム・コントロール（通称OCX）を発表した。 これはOLE2.0 のオートメーションの機能を使ってカスタム・コントロールを実現するものである。使い方はVBXと同じであるが、Visual Basic 以外でも使える。今後、VBAやVisualC＋＋でもOCXが使えるようになり、WINDOWS NT や Chicago の 32 ビット環境用のOCXも登場することになる。

2．3 マルチメディア

近年ハードウエア、ソフトウエアのめまぐるしい進歩により、オーディオ、イメージ、動画といった個々の メディアを統合したいわゆるマルチメディアをPC上で扱うことが可能になった。1995 年現在の時点ではマルチメディアを扱った情報システムの構築は、まだ本格的に開始されたとは言いがたいが、マルチメディアがもつユーザーインタフェースへの有効性、すなわち文字だけで表現できない情報をユーザーへ提供することが可能であるというメリットを否定する人はおそらくいないであろう。

こうしたメリットを期待できるマルチメディアであるが、システムを開発する立場から見ると、文字等に比較して著しく複雑なデータを扱う必要があること、そして技術の進歩と共に変遷

するデータ形式に対して継続的に対応しなければならないこと、さらに今後次々 と出現が予想される新しいメディアへも適切なタイミングで対応しなければならない等といった課題がある。こうした課題に対しデータとそれを扱う操作を別々 に扱う従来の開発方法で対応していては、とてもユーザーが満足する開発期間やコストでシステムを構築することは不可能である。

それに対しオブジェクト指向技術はかなり有効な解決策を提供する。すなわちカプセル化とポリモーフィズムである。

マルチメディアを扱うアプリケーションにおいてカプセル化は、動画などを記録した複雑なデータとその操 作方法を切り離さずに一つのオブジェクトとして実装することを意味する。カプセル化によりシステム開発者は、自らAPIを開発することなく様々なデータを 扱うことが可能となる。

また、動画の表示を開始するのも音楽の演奏を開始するのも、開始という意味は同じである。この例をオブ ジェクト指向的に解釈すると、「動画オブジェクトに開始メッセージを送ると動画の表示が開始され、音楽オブジェクトに開始メッセージを送ると音楽の演奏が 開始される」となる。このようにオブジェクトの種類が異なっても、同じ意味の操作に対して、同じAPIを適用することをポリモーフィズムと呼ぶ。ポリモー フィズムはプログラムの構造を単純にし、さらに新たなメディアの組み込みも容易にする。

以上のようにマルチメディアに対しオブジェクト指向技術は大変有効である。したがってシステム開発部門では今後マルチメディアが広まると共にオブジェクト指向技術の修得が不可欠となっていくであろう。

2．4 クライアント／サーバーシステムの普及

ここ数年の情報システム化の動向で特徴的なものに、クライアント／サーバーシステムの普及が挙げられる。ここではクライアント／サーバーシステムの普及とオブジェクト指向技術の関係について見ていきたい。

（1）システム開発の容易性

現在普及しているクライアント／サーバーシステムの多くはGUIを提供するクライアントモジュールを PC上に配置し、データを管理するサーバーモジュールをWS上に配置するタイプであろう。こうしたクライアント／サーバーシステムが普及しはじめた当初に は、クライアント／サーバーシステムのメリットとしてダウンサイジングによるハードウエアコストの削減がもっぱら強調されていた。

しかしながらシステムに対する要件が複雑化してくるに従って、システム開発におけるソフトウエアのコス トがますます増大してきており、相対的にハードウエアのコストが占める割合は、小さくなっているのが現状である。システム要件の複雑化の傾向は今後も続く ものと思われるため、企業にとってソフトウエアコストの削減は至上命題となっている。

こうした背景をふまえ、ソフトウエアコストの削減に対して、クライアント／サーバーシステムが果たす役 割を考えてみたい。クライアント／サーバーシステムは、GUIを提供するクライアントモジュールと、データを管理するサーバーモジュールを物理的に分離 し、なおかつ自らのAPIのみを相手に公開し、内部のロジックを隠蔽していることが特徴である。そのためクライアント／サーバーシステムでは、クライアン トモジュールのロジックやGUIのレイアウトを変更しても、APIにさえ変更がなければサーバーモジュールは何も修整する必要がない。こういった内部ロ ジックを外部から隠蔽する考え方はオブジェクト指向のカプセル化の概念にも一致しておりシステムの保守性を向上させるためには不可欠なものである。言い換 えるなら、クライアント／サーバーシステムはオブジェクト指向のカプセル化の概念に基づいたシステムであり、そのためカプセル化のメリットとして、ソフト ウエア開発コストの削減が期待できるということである。

2．1節でも取り上げたように、ビジネスを取り巻く環境の変化が早い今日では、その変化に対してシステ ムを素早く追随させることがそのまま企業の競争力の向上となってきている。そういう意味で保守性の向上が期待できるクライアント／サーバーシステムは今後 ますます注目されるシステム形態となるであろう。

（2）プロトタイプツールの充実

今日のクライアント／サーバーシステムの普及を加速しているものの一つにプロトタイプツールが挙げられ る。現在主流を占めているプロトタイプツールの多くは、GUIをサポートし、さらにサーバー上のRDBへのアクセスも視覚的に定義できるものでビジュアル プログラミングツールと呼ばれている。これらのツールの多くはオブジェクト指向技術をベースに開発されており、このツールを用いることで通信やRDBの専 門的な知識がなくても、容易にクライアント／サーバーシステムを構築することが可能となり、その普及を後押ししている。

（3）次世代のクライアント／サーバーシステム

現在開発されているクライアント／サーバーシステムの多くは、クライアント側にGUIとデータアクセスロジックを持ち、サーバー上のRDBをリモートアクセスするいわゆる2－Tier の構造である（図2－4－1参照）

図２－４－１　Tireアーキテクチャ

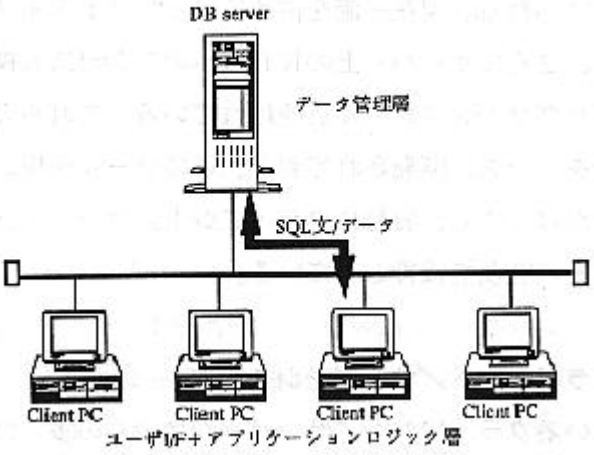


図２－４－１

しかし、前述のようにビジネス環境の変化への追従性が重要になってくると、データアクセスロジックとビジネスロジックをそれぞれ物理的に独立したモジュールとしてサーバー側で管理し、保守性とデータの整合性を向上させた３－Tier（図２－４－２参照）のシステムの必要性が指摘されるようになってきた。それに伴い、異種分散環境でこのようなシステムを支える基盤技術の標準化がOSFやOMGで進められており、活動の成果としてDCE（Distributed Computing Environment）やCORBA（Common Object Request Broker Architecture）といったミドルウエアの仕様が公開されてきている。

図２－４－２　Tireアーキテクチャ

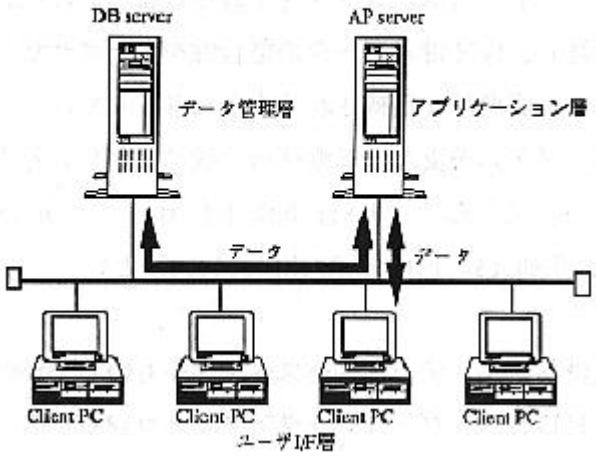


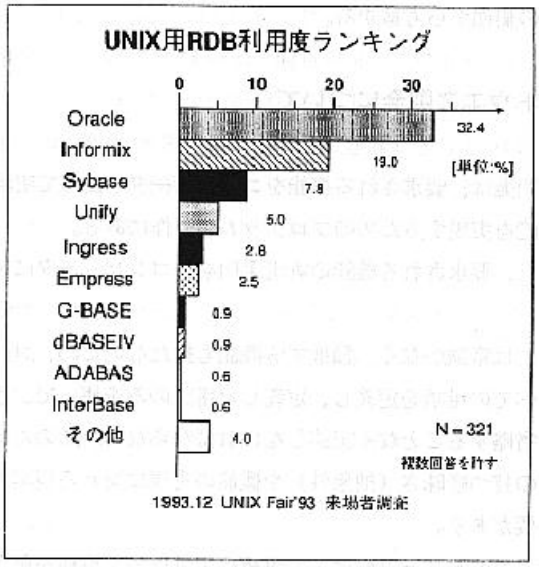
図２－４－２

このような背景のもとに、データベースベンダーもいくつかの仕組みを提供してきている。例えば、RDB ベンダーはデータアクセスロジックそのものをデータベースで管理するストア

ドプロシジャという機能を提供してきており、次世代のデータベース管理システム として期待されているオブジェクト指向データベースベンダーもデータだけでなくメソッドそのものを管理の対象とするアクティブデータベースを既に市場に投入してきている。この2つのタイプに共通するのは、今までのようにデータとそれをアクセスするプログラムを分離して管理するのではなく、それをひとまとめ にして管理しようとしている点である。このことは、オブジェクト指向のカプセル化の考え方に一致しており、クライアント／サーバーシステムが今後オブジェクト指向の考え方をますます取り入れていくであろうことを暗示している。

図2－4－3は、Unix Business Association 発行の Open Directory 1994 年版から抜粋したRDBとOODBの市場動向である。若干古いデータであるが、参考にして頂ければと思う。

図2－4－3



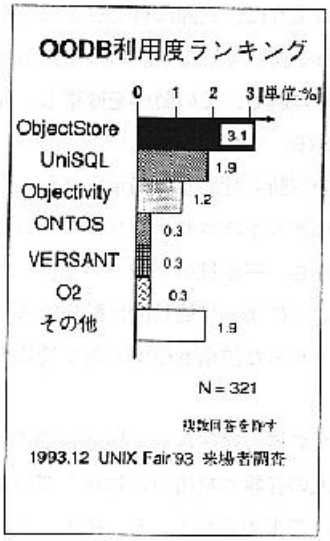


図2－4－3

第3章　なぜオブジェクト指向なのか

本章では、いま、なぜオブジェクト指向が注目されているのかについて、ソフトウエアの開発作業の側面から考察する。

3．1 ソフトウエア開発について

ソフトウエア開発は、要求される機能をコンピュータを使って実現する作業であり、開発の対象は機能を実現するためのプログラムの製作にある。

プログラムとは、要求される機能の実現手順を、コンピュータに示す手順書であるとも言える。

コンピュータには常識がなく、類推する機能も持たないため、コンピュータに手順を示すには、すべての用語を定義し、定義した用語のみを使って、手順を構成 するすべての指示を、省略することなく記述しなければならない。このため、プログラムの製作には、言葉の持つ曖昧さ（抽象性）や機能の実現に関わる現実の 複雑さという問題を解決する必要がある。

プログラムは、コンピュータによって現実の電気信号や機械を動かしているにもかかわらず、他の工業製品のように、実際に目で見ることによって機能・品質・ 完成度などを認識できるような形を持っていない。形のないものは直観的にとらえにくく、このことが、プログラムを理解し、その動作を観察し、品質や機能を 評価するということを難しいものにしている。

またコンピュータは何の判断も行わずに指示されたことを確実に実行するため、コンピュータに示す手順には誤りが許されず、プログラムには完全性が要求される。

複雑さ・抽象性・形のなさ・完全性の要求という、プログラムの持つ問題点は、組織や方法論、開発ツールなどによって機械的に解決できるものではなく、問題の解決には技能や経験、センスといった技術者の個人的な技術力が現状では不可欠の要素となる。

したがって、これら言葉の持つ曖昧さや、機能の抽象性の解決や、形のないソフトウエアの把握といった個人の経験や技術力に依存している問題を解消できる開 発手法がソフトウエア開発において求められている。そうしないと、いつまでたっても人に依存した開発が続くことになるからである。

3．2 オブジェクト指向とは何か？

曖昧で、複雑で、目に見えにくいというソフトウエアの根本問題を解決し、明確で、単純で、理解、把握の容易なソフトウエア開発を可能にする考え方として、「モデル化」と「部品化」がある。曖昧でかつ複雑なソフトウエア対象の現実世界も、うまくモデル化できれば、単純で、世界の様子が容易に理解できるように なる。また、ソフトウエアがすでに内容がよく理解できている部品の再利用、組み合わせによって構築できれば、ソフトウエアの構成、動作は格段に単純化で き、その信頼性も桁違いに向上する。

ソフトウエア開発の問題点を解決する「モデル化と部品化」の考え方は、まさにオブジェクト指向開発方法論のキーとなる考え方になっており、オブジェクト指 向が注目を集めている理由と言える。以下では、オブジェクト指向のキーとなるモデル化と部品化を容易に実現するクラスの概念について説明する。

(1)モデル化について

オブジェクト指向の開発方法論では、開発対象分野のモデル化をその対象分野の直接の対象物を部品化対象(=オブジェクト)としてとらえ、その相互関係、相 互作用を現すことで行う。例えば会計システムでは、店舗や事務所、従業員がオブジェクトとして部品となる。このように、現実世界で実感できる対象をそのままモデル化しておけば、またモデル化の通りプログラムを構成しておけば、それを構成する個々の部品(=オブジェクト)が何か、何をするのか、また、そのプ ログラムがどう構成されているのかが、誰にでも直観的に理解しやすいものになるということである。

プログラム部品を共有し再利用するための最も重要な条件は、その部品が何なのか、何をするのかがわかりやすいということである。個々の部品が理解しがたい ものであれば、必要とする部品を見つけることすらできないし、使いこなすことはさらに困難であろう。プログラムを効率よく開発するために部品化は重要な概 念であるが、部品化、再利用をうまく行うためには、モデル化がそれにもまして重要であると言える。オブジェクト指向の開発方法論では、分析、設計の段階で このモデル化を行う。

(2)クラスについて

オブジェクト指向ではプログラミングされた部品をクラスと呼ぶ。クラスはオブジェクト指向による部品の再利用等の生産性を高めるための多くの基本的な特徴を持っている。

ここでは下記のような“クラス” というプログラム部品の形式の特徴を表す基本的な機能のいくつかと、その意味について説明する。

・ カプセル化 プログラム部品の内部へのアクセス方法を制限し、個々の部品の独立性を高める方法
・ 継承　　複数のプログラム部品に同じ性質を共有させる方法
・ 多態　　1つの操作指示で複数の部品に同様の要求を許す方法

1)カプセル化について

カプセル化とは、プログラム部品の外部からのアクセス方法を制限し、個々の部品の独立性を高める手法だと言える。
プログラム部品の独立性を高めなければならない理由は、下記の2点にある。

①変更の影響を小さくする
②部品を再利用しやすくする

それでは、なぜアクセス方法を制限すると変更の影響が小さくなり、部品を再利用しやすくなるのであろうか？

プログラムは、コンピュータへの指示を示す手順書であると言える。つまり、指示を集めた文書であるから、同じ指示の記述がどこかにあるのならば、何度も同じことを書かずに、その記述を参照するようにした方が、統一性のあるプログラムを楽に作成することができる。

問題は、どういう参照方法許すかということにある。

たとえば、記述の単位を無視した参照方法（例：144 ページの 17～20 行目というようなやり方）を許したとすると、その部分やその前後の文字を1文字で も変更したならば、その部分を参照しているすべての箇所に影響が出るかもしれないし、少なくともその部分を参照している箇所に、影響がないかどうか調べる 必要がでてくる。

また、記述内容の意味を示さない参照方法（例：2章3節4項というようなやり方）は一般的な方法であるが、変更への対応や再利用という面では、良くない方 法だと言える。なぜなら、文書の構成を変更しようとする場合や別の文書で参照している部分を使おうとする場合に、もろにその影響を受けてしまうことになる からである。

したがって、同じ指示の記述への参照は、“記述単位をその意味する名称によって行う”（例：店舗.会計.月報作成）というように参照方法を制限することにより、記述単位の中に変更があった場合でも、文書の構成に変更があった場合でも、その記述単位を参照している箇所には、何の影響も生じないようにできるわけである。

2)継承について

会社には、役員・管理職・一般職などの様々な役割や権限、職務を持つ人々が存在する。
社員管理システムを考えた場合、これらの人々を1つの型のデータで扱うのは、大きな無駄を生みバカげている。したがって、社員管理システムでは、役員ならば役員用の、管理職ならば管理職用のデータ型を定義して、プログラムを記述することになる。
これまでのプログラミング言語では、データの型に合わせて処理が行われるように、人間が

考えてプログラムを記述しなければならなかった。
たとえば、全社員の給与計算を行う処理は、下記のようになった。

ところで、役員も管理職も一般職も、社員としての共通の性質を持っており、給与計算もその一部に入る。オブジェクト指向では、共通の性質は〝継承〟という方法により共有させることができるため、同じ処理を下記のように記述することができる。
これらの2つの処理を比べれば、オブジェクト指向によりいかにプログラムの生産性を向上させることができるかを理解していただけると思う。

3)多態について

日本語のように、われわれが日常使用する自然言語では、1つの言葉を複数の対象に利用することができる。
たとえば、社員に関する情報を印刷する場合、下記のようになる。

a.“役員情報〟を“印刷〟する
b.“管理職情報〟を“印刷〟する
c.“一般職情報〟を“印刷〟する

つまり、処理対象の情報の種類に関係なく、印刷を行うには“印刷〟という言葉を使う。
オブジェクト指向言語では、プログラムをより自然に記述できるようにするため、“印刷〟という同じ言葉を、処理対象を越えて使用することを許している。
ところで、これまでのプログラミング言語では多態の機能がないため、一つの言葉は一つの意味にしか使えない。たとえば、社員情報を印刷しようとする場合、印刷対象別に複数の印刷処理を表す言葉を使わなければならなかった。
これまでのプログラミング言語での印刷処理の記述は、下記のようになる。

a.“役員情報〟を“役員情報印刷〟する
b.“管理職情報〟を“管理職情報印刷〟する
c.“一般職情報〟を“一般職情報印刷〟する

この例をみれば、多態の機能のないプログラミング言語では、プログラムの記述が無駄が多く不自然であることが理解していただけると思う。

3．3 オブジェクト指向に関する用語解説

オブジェクト指向に関する基本的な用語の解説を行う。大きくは、次の5つのタイトルに分けて説明する(図3－3－1参照)。

・ オブジェクト指向の基本概念
・ オブジェクト指向プログラミング
・ オブジェクト指向データベース
・ オブジェクト指向方法論
・ 分散オブジェクト環境

表3－3－2　OODB製品

分類（開発の経緯）		製品名	開発
プログラミング言語で開発したオブジェクトの永続化	C++ベース	Objectivity/DB	米Objectivity社
同上	同上	ObjectStore	米ObjectDesign社
同上	同上	ONTOS	米Ontos社
同上	同上	PERCIO	NEC
同上	同上	VERSANT	米Versant Technology社
同上	Smalltalkベース	GemStone	米Servio社
複雑な構造を持つデータモデルのサポート（RDB拡張型など）		MATISSE	仏A.D.B社
同上		O2	仏O2 Technology社
同上		ODB2	富士通
同上		Prsistence	米persistance Soft社
同上		UniSQL	NTTデータ通信、米UniSQL社

図3－3－1

(1)オブジェクト指向の基本概念

オブジェクト指向では、データと手続きを一体化したオブジェクトと呼ばれる実体のモデルが、プログラ ミングの基本単位となっている。オブジェクトを動作させるためには、オブジェクトにメッセージが送信すればよい。オブジェクトの中でも、類似の機能をもつ オブジェクトの集めたものがクラスとなる。クラスは、従来プログラミングの「型」の拡張に相当するもので、オブジェクトは、すべてクラスから生成される。 そのクラスの階層構造と継承機能により、既存のメソッドの再利用が可能となる。

・オブジェクト(Object)

コンピュータを利用した問題解決を行うためには、現実世界の問題をなんらかの形でモデル化する必要 がある。この現実世界の実体を計算機の中にモデル化する際の単位として、その実体の特徴（属性）と振る舞い（機能）とを一体化したモデルがオブジェクトで ある。つまり、オブジェクトとは、特徴であるデータとその振る舞いであるメソッドとを持つ、実体のモデルである（図3－3－2）。

図３－３－２　オブジェクト

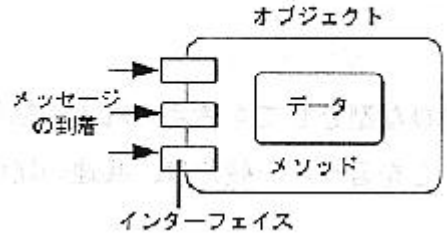


図３－３－２

・メッセージとメソッド（Message, Method）

メッセージは、オブジェクトを動作させるためのリクエストである。オブジェクトの状態を変更するためには、そのオブジェクトに対してメッセージを送信するという方法をとる。受信されたメッセージをどのように処理、操作するかは、そのメッセージを受信したオブジェクトが判断して実行する。

メソッドは、オブジェクトに内包されている手続きである。オブジェクトがメッセージを受信した場合、そのメッセージの内容をメソッドがどのように処理するかを決定している。

・メッセージパッシング（Message passing）

メッセージパッシングとは、オブジェクトに対する処理の要求が、メッセージによるメカニズムで実現されていることを言う。図３－３－３ではオブジェクト2が “print”というメッセージをオブジェクト1に対して送信し、オブジェクト1は、そのメッセージに対応したメソッドを内部で起動し、さらにその結果を“finished” というメッセージでオブジェクト1に送信している仕組みを示している。

図３－３－３　メッセージパッシング

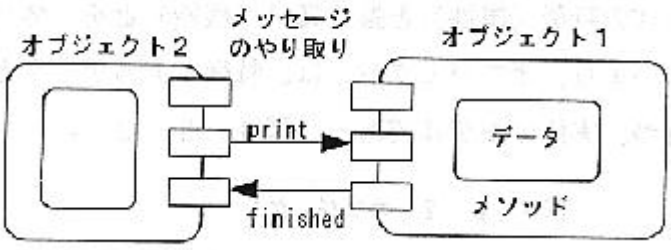


図３－３－３

・クラスとサブクラス（Class, Subclass）

クラスとは、オブジェクトに共通の属性や手続きをひとまとめにした単位のことを言う。実行時では、 クラスを実体化して作られるのが「オブジェクト」であるので、クラスは、オブジェクトのひな型としても考えられる。抽象クラスとは、オブジェクトを生成し ないクラスのことを言う。一般には、共通の属性やメソッドだけを持たせるために存在する。

サブクラスとは、クラス階層間で、注目するクラスの子（下位）のクラスのことを言う。クラスを作成する場合、まず、その言語環境が提供するクラス群を参 照し、その機能や構造が適切なクラスを親クラスとして、そのクラスのサブクラスとしてクラス定義することが多い。その際、クラスの作成能力は、実体をどの ようにモデル化するかということと、その言語環境でどのようなクラスがあるかを即座に判断できることによる。

クラス作成の操作には、大きくは2つある。複数のクラスに共通なクラスとして親クラスを定義する操作を汎化（Generalization） と言う。また、あるクラスから、特定の機能や構造に着目し、そのクラスの子クラスを定義する操作を特化（Specialization） と言う。

次に、Smalltalk-80 の集合関連のクラス階層を例示する（図3－3－4）。

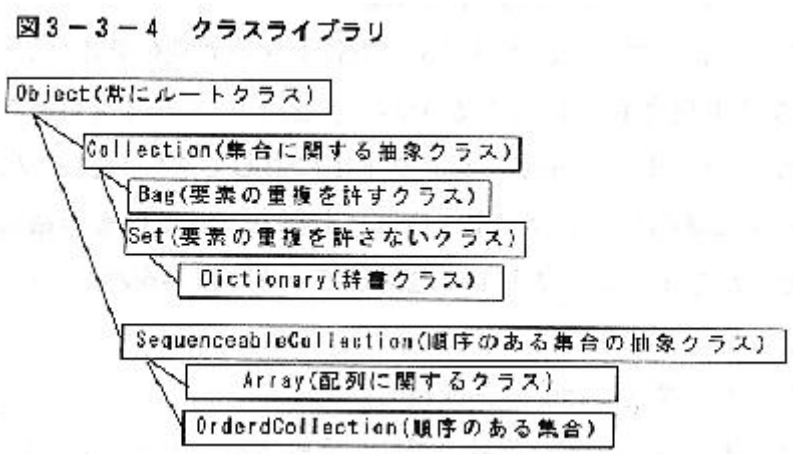


図3－3－4

・継承（Inheritance）

上位クラスの属性とメソッドを、下位のクラスが引き継ぐことを継承と言う。継承のメカニズムにより、一部の機能だけの追加や修正部分だけのコーディング作業でプログラミングを行う差分プログラミングが可能となる。

図3－3－5に、Document クラスを親クラスとし、Text, Paragraph のサブクラス の階層関係を例に示す。Paragraph クラスから生成するオブジェクト(aParagraph Object） のインスタンス変数には、その親クラスのインスタン変数である page、 pageSize、 font、 style が継承されている。またメソッドである print、saveFile、 justify、copy も継承されている。

図３－３－５　インヘリタンス

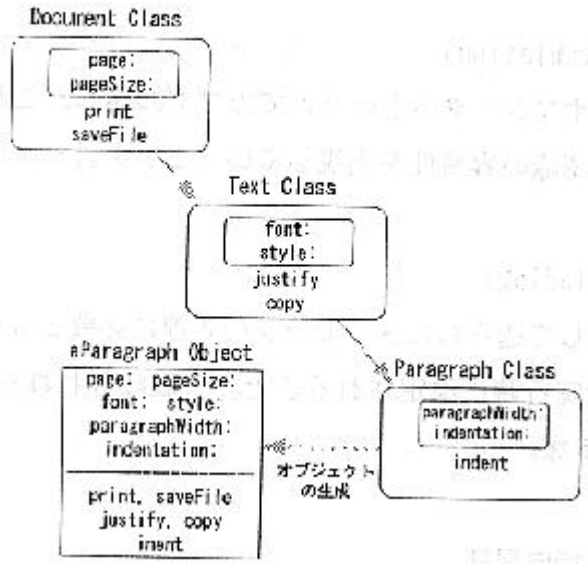


図３－３－５

・ポリモーフィズム（polymorphism）：多態

ポリモーフィズムとは、同じメッセージであっても、そのメッセージの送信オブジェクトにより、処理が異なることを言う。その結果、ソフトウエア開発者は、種々のオブジェクトに対して同一のメッセージによる、処理ロジックの簡素化、共通化ができる。

図３－３－６では、Image クラスの anImage Object と Document Class の aDocument Object に同一のメッセージ“saveFile”を送る様子を示している。

図３－３－６　ポリモーフィズム

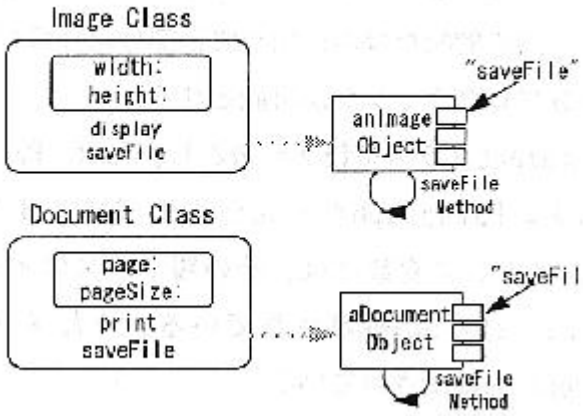


図３－３－６

・カプセル化（Encapsulation）

データと手続きをオブジェクトという形でカプセル化し、これにより、抽象データ型、情報隠蔽機能、認識の容易性を実現していることを言う。

・遅延結合（late binding）

オブジェクトに対して送られたメッセージの処理に必要となる手続きが、オブジェクトの型に応じて、実行時に決定されること。一般に言われる、動的結合（dynamic binding）と同様の意味。

（2）オブジェクト指向言語

オブジェクト指向言語は、従来のプロセスを中心とした手続き型言語とは違い、実体モデルであるオブジェクトを記述する言語として発展してきた。大きくは、Algol をルーツとするシミュレーション言語から発達した流れと、独自にAI（人工知能）の研究としてリスト処理用の言語（LISP）から発達した流れ がある。その発展の歴史の中でも、オブジェクト指向の概念を明確に言語として機能化したものが Smalltalk である。クラスやインスタンス、インヘリ タンスやポリモーフィズムなどの概念を実装し、GUI、ブラウザおよびクラスライブラリなどの利用が可能であるので、初期の頃には、研究目的やプロトタイ プ開発では、十分な実績をあげてきた。Smalltalk の登場により、各種の言語（C, Pascal）のオブジェクト指向化が加速された。

最近は、4GLを言語ベースとしたアプリケーション開発環境が、部品化の概念と部品の再利用を取り込み、製品（SQL-Windows, PowerBuilder など）化されている（図3－3－7）。

図3－3－7　オブジェクト指向言語の歴史

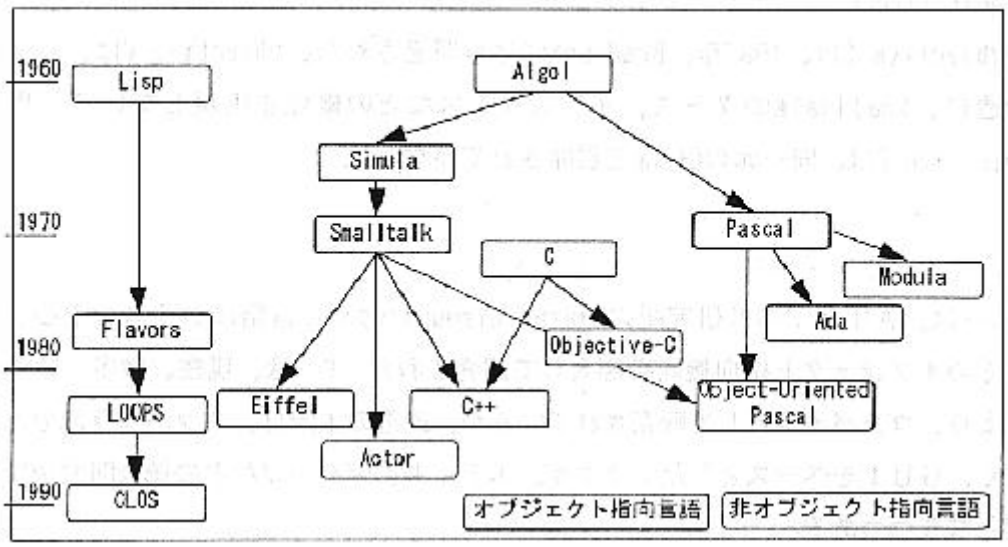


図3－3－7

・シミュレーション言語　Simula

Simula は、1960 年代、Algol にカプセル化や継承の概念を加えて、システム記述言語として開発された。現行のほとんどすべてのオブジェクト指向言語は、Simula をルーツとして発展してきた。

・Smalltalk-80

ゼロックスのパルアルト研究所で、Adele Goldberg を中心として開発された言語。1970 年代当時では、ビジュアルインタフェースやビットマップエディタ、クラスブラウザなどの現在のオブ ジェクト指向プログラミング環境のベースとなるものを最初から備えていた。
Smalltalk-80 は、オブジェクト指向プログラミングの概念を、オブ ジェクト、クラス、メッセージ、インスタンス、メソッドの5つの言葉で明確に定義し、その功績は非常に大きい。
現在では、クライアント／サーバー型のアプリケーションの開発環境として、データベース接続やビジネスチャートなどのクラスライブラリとGUI環境を備えたオブジェクト指向統合環境 Visual Works として販売されている。PC、WS、Macintosh の各種のプロットホームで、ソースファイルのバイナリ互換が実現されているので、開発環境と運用環境を意識することなく、開発に専念できる。

・Objective-C

Objective-C は、1987 年、Brad Cox により開発された。Objective-C は、C言語の 構造に、Smalltalk のクラス、インスタンスなどの機能を実現している。現在、 Objective-C は、同一派の C++ほど展開されていない。

・C++

C++は、AT＆Tベル研究所の Bjarne Stroustrup がC言語にクラスを中心とする多くのオブジェクト指向機能を導入して開発された。C++は、現在、数多くのメーカーより、コンパイラ として販売されているが、最近の主流は、コンパイラ部分だけでなく、GUIをベースとしたブラウザ、エディタ、デバッガなどの統合開発環境が主流となりつ つある。

・Visual C++

Windows ベースのアプリケーションを開発するためのフレームワークを提供する統 合開発環境。マイクロソフトが従来、開発者に提供してきたSDK（Software Developer's Kit)の他に、オブジェクト指向の概念を導入したMFC(Microsoft Foundation Class）が利用可能である。この2つのリソースライブラリを用いて Windows アプリケーションの開発を柔軟に行うことが可能となる。

・SQL-Windows

SQL-Windows は、米国 Gupta 社が開発している、ビジュアルなオブジェクト指向プログ ラミング環境。基本的なクラスは、実際の画面や表示系に関るGUIクラスと、データ型やモデルそして手続きに関わるファンクショナルクラスがある。データ ベースは、RDBをターゲットとしており、そのための呼出しや参照の手続きが非常に容易に記述できるようになっている。クラスの宣言やメソッドの記述は、 SAL（SQL-Windows Application Language） という、フル4GLで記述するようになっており、そのためのSAL専用のアウトラインエディタが用意されている。

（3）オブジェクト指向方法論

伝統的な機能分割によるソフトウエア開発は、システムのもつ抽象的な機能を決定し、さらにそれを細分化 していくという、トップダウン的な方法に基づいている。オブジェクト指向によるソフトウエア開発は、まず、現実の具体的な実体である物に着目し、その物を 中心としてオブジェクトとしてモデル化し、そのオブジェクトの関係および組み立てによりソフトウエアを開発するという、ボトムアップ的な方法に基づいている。

オブジェクト指向によるソフトウエア開発ライフサイクルは、従来と同様、企画、分析、設計、構築、保守というフェーズを持つ。しかし、従来の後戻りを許さないような（仕様の変更が困難、構築までに時間を要する）ライフサイクルと異なり、オブジェクトという理解容易な単位を媒体として、企画者、分析者、設 計者、構築者、保守者そして利用者の間でのコミニケーションを円滑に行うことにより、現フェーズからすべてのフェーズへの参照が容易な形態になっている。 この形態は、従来のウオーターフォールモデルに対して、海洋（オーシャン）モデルと呼ばれている（図3－3－8）。

図３－３－８　従来型とオブジェクト指向型

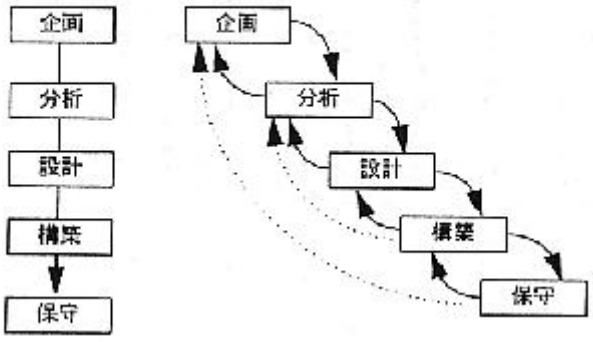


図3－3－8

現在、オブジェクト指向方法論として手法化されている代表的なものには、次の4つがある。

－ OMT Method
－ Booch Method
－ OOA/ OOD Method (Coad & Yourdon)
－ OOSA Method (Shlaer & Mellor)

表3－3－1に、これら4手法の比較を示す。また、すでにこれらの手法を搭載した各種のCASEツールが開発・販売されている。

表3－3－1

表3－3－1　主要4手法の比較（OMT、OOA／OOD、Booch、OOSA）

	OMT	OOA/OOD	Booch	OOSA
開発者	J. Rumbaugh 他	Coad　Yourdon	Grady Booch	Shlaer　Mellor
発表年	1991	1990	1990	1988
サイクルのサポート	分析、設計、構築	分析、設計、構築	設計	分析
特徴	・オブジェクトモデル、動的モデルおよび機能モデルの3つのモデルにより情報モデルを記述する 分析、設計、構築までスルーする手順が確立されている ・言語別の実装に関する書籍などガイドラインとしも完成度が高い	・設計から少数のオブジェクト指向概念に基づく記法の設定とモデル構築により、利用者が理解しやすい ・オブジェクト間の関連や振る舞いに関する表現力が乏しい ・現場でユーザとのオブジェクトの抽出や整理に効果的である	・主要なオブジェクト指向言語へのマッピングをサポートしている ・多種の図を使用するので、記述能力が高い。その反面、設計が複雑になる	・オブジェクトの関係表現が豊富 ・関係自体もオブジェクトで表現 ・大規模リアルタイムに対しても適用可能な手順の体系にがなされている
問題	手法の習熟に期間を要する	動的モデルの記述能力が弱い	分析フェーズのサポートが弱い	設計への連続が弱い
記述モデル1	オブジェクトモデル ・オブジェクト図	オブジェクト図	静的モデル ・クラス図 ・オブジェクト図	情報モデル ・情報構造図　・オブジェクト仕様書 ・関係仕様書
記述モデル2	動的モデル ・状態遷移図　・シナリオ ・イベントフロー、トレース図	動的モデル ・オブジェクト遷移図	状態モデル ・状態遷移図　・タイミング図	状態モデル ・状態遷移図　・イベントリスト ・オブジェクト通信モデル
記述モデル3	機能モデル ・データフローダイアグラム	サービスチャート	テンプレート	プロセスモデル ・アクション・データフロー・ダイアグラム 状態プロセス表　・プロセス記述
備考	現在、オブジェクト指向方法論の中で、最も普及している手法である	構造化分析・設計手法と互換のある Yourdon により記述された手法である。手法自体は、記述でわかりやすいのでオブジェクト指向分析の入門として最適	元来は、Ada 言語のための設計手法として始まり、オブジェクト指向へと拡張された。その意味で、アプリケーション開発での事例は数多い	大規模なリアルタイム系のシステム開発で用いられてきた設計手法のオブジェクト指向に基づいた体系化
製品サポート	Sto/CME 他数種	Object Cost 他数種	Rose　ParadigmPlus	Teamwork

(4)オブジェクト指向データベース(Object Oriented Database)

1989 年 12 月に京都で開催された第1回演繹・オブジェクト指向データベースシステムに関する国際会 議(DOOD89：Deductive and Object Oriented Database) で、F.Banchilhonらは、「オブジェクト指向データベースシステム宣言」という論文で、オブジェクト指向データベースシステムが具備すべき条件 として、13 の必須条件と5つの付加条件をあげている。

・ OODBの 13 の必須条件

「オブジェクト指向データベースシステム宣言」では、あるDBMSがオブジェクト指向データベースであることの必要条件として次の 13 の基本機能をあげている。

－複合オブジェクト(complex object)
－オブジェクトの識別性（object identity）
－カプセル化（encapsulation）
－型またはクラス（type, class）
－継承（inheritance）
－遅延束縛（late binding）と一体化した再定義（overriding）
－拡張可能性
－計算の完全性
－永続性
－二次記憶管理
－並行処理制御
－障害回復
－アドホック（ad hoc）な問い合わせ（query）

・ **複合オブジェクト（complex object）**

複合オブジェクトとは、より単純なオブジェクトから構成されるオブジェクトのことを言う。例えば車のオブジェクトの関係をあげる。図3－3－9に 示す車 （Vehicle Object）は車体(Body Object)、製造メーカー（Manufacturer Object)、色（Color Object）、エンジン(Engine Object)から構成されている。これらの各構成要素となるオブジェクトと元の車(Vehicle Object）とは、部分－全体(part-of)という関係にある。このように、ある1つのオブジェクトが他のオブジェクトの構成要素から成り立っているとき、その1つのオブジェクトを複合オブジェクトと言う。

図３－３－９　複合オブジェクト

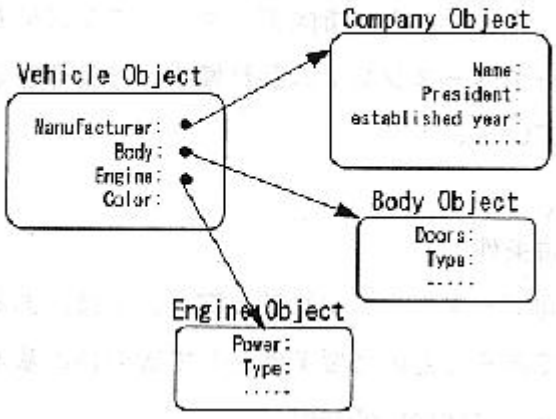


図3－3－9

・ **市販されているOODB製品**

OODBには、大きく、言語で作成したオブジェクトに永続性を持たせるためのデータ格納庫としての発展と、RDBからの複雑なデータタイプを扱う方向の発展がある。現在日本で市販されている各種のOODB製品の一覧を表3－3－2に示す。

表3－3－2

表3－3－2　OODB製品

分類（開発の経緯）		製品名	開発
プログラミング言語で開発したオブジェクトの永続化	C++ベース	Objectivity/DB	米Objectivity社
同上	同上	ObjectStore	米ObjectDesign社
同上	同上	ONTOS	米Ontos社
同上	同上	PERCIO	NEC
同上	同上	VERSANT	米Versant Technology社
同上	Smalltalkベース	GemStone	米Servio社
複雑な構造を持つデータモデルのサポート（RDB拡張型など）		MATISSE	仏A.D.B社
同上		O2	仏O2 Technology社
同上		ODB2	富士通
同上		Prsistence	米persistance Soft社
同上		UniSQL	NTTデータ通信、米UniSQL社

（5）分散オブジェクト指向環境（Distributed Object-Oriented Environment）

OMG（Object Management Group)は、1989 年に“オブジェクトマネージメント”という技術の標準化団体として発足した。現時点では 400 社近い会員企業を擁する。

- ORB （Object Request Broker）

ORBとは、オブジェクトが処理要求を発行し、その応答を受け取れる透過的なメカニズムのことを言う。この機能により、異機種に分散したマシン上のアプリケーションのインターオペラビリティが可能となり、複数のオブジェクト・システムがシームレスに結合される。

- CORBA （Common Object Request Broker Architecture）

CORBAとは、OMG （Object Management Group） により、採択された共通のORB技術のことを言う。CORBAが定める共通のORBサービスとインタフェースの提供により、異なったORBのインプリメン テーション間において、クライアントとオブジェクトのインプリメンテーションのポータビリティの確保が可能となる。 この共通仕様は、主に、DEC、 HP、NCR、HyperDesk、Object Design、 Sun Soft の6社による共同提案により生まれた。

・ 分散オブジェクトモデル (Distributed Object Model)

オブジェクト指向の基本的考え方に、「オブジェクト間でのメッセージのやり取り」がある。つまり、 クライアントからオブジェクトへのサービスのリクエスト(要求)という形のモデルである。このモデルは、複数の接続された計算機環境においても拡張可能な モデルであり、分散環境でのサービスを行うオブジェクトへの、クライアントからの要求というモデルとなる。この分散環境に拡張されたオブジェクトモデルを分散オブジェクトモデルと言う(図3－3－10)。

図3－3－10 分散オブジェクトモデル

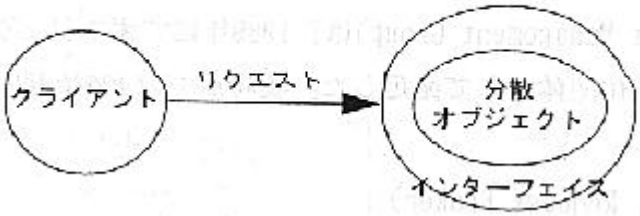


図3－3－10

・ OMAのリファレンスモデル(Object Management Architecture , Refernce Model)

OMAのリファレンスモデルは、ORB、オブジェクトサービス(Object Service)、共通ファシリティ (Common Facility)、そしてアプリケーションオブジェクトにより構成される。ユーザーが直接関係するのは、アプリケーションオブジェクトであり、ユーザー のオブジェクト・モデルを実現したオブジェクト群である。アプリケーションオブジェクトは、通常のアプリケーションやプリンタやFAXなどをモデル化した オブジェクトを含む。他の3つは、このオブジェクトモデルを実現するためのツールである。そして、この3つの部分のアーキテクチャについて、OMGは標準 案を作成、提供している(図3－3－11)。

図3－3－11 OMAリファレンスモデル

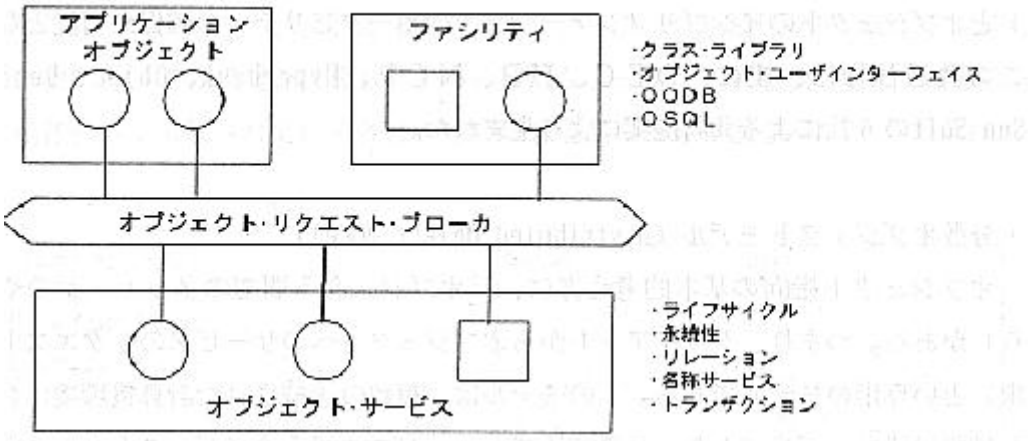


図3－3－11

〔参考文献〕

□オブジェクト指向の基本概念について

・ Object-Oriented Software, Ann L. Winblad 著、Addison-Wesley Publishing,1990
・ オブジェクト指向実用期へ、日経インテリジェントシステム、別冊 1992 春号、1992・オブジェクト指向技術総覧、93-94 年版、日経インテリジェントシステム別冊、1994
・ オブジェクト指向のシステム開発、本位田　真一、山城　明宏　著、日経BP出版センター、1993
・ オブジェクト指向設計、酒井　博敬、堀内　一　著、オーム社、1994

□オブジェクト指向方法論について

・ オブジェクト指向方法論OMT、モデル化と設計、J.ランボー他著、トッパン、1992

□オブジェクト指向データベースについて

・ オブジェクト指向データベースシステム宣言とその意義、西尾　章治郎、田中　克己、Bit, Vol.22, No.8
・ オブジェクト指向データベース、基礎知識から製品の評価まで、日経インテリジェントシステム別冊、1994
・ インテリジェントCAD、CAM、CAE研究会、第6回研究会資料、田中　克己、1992 年3 月 12 日

□分散オブジェクト環境について

・ 共通オブジェクトリクエスト・ブローカ　～構造と仕様～　オブジェクトマネジメントグループ編著、相磯　秀夫　監訳、社団法人日本オフィスオートメーション協会、1992

第4章　オブジェクト指向Q＆A

本章では、本部会の活動を通じ、オブジェクト指向に関する検討を行った際に、導入や、教育の問題、開発　の方法など、メンバーからあげられた様々な疑問を取り上げる。オブジェクト指向は、事例も少なく、当初はメンバーも理解に苦しんだテーマであったが、これ　ら疑問点を分類し、検討していくうちに、以下のような解答を導き出すことができた。「オブジェクト指向」導入を考えられているユーザーの方々の参考になれば幸いである。

4．1　オブジェクト指向の導入

Q. 最近「オブジェクト指向」という言葉をよく耳にしますが、実際にこの方法で開発を行ったという話は聞きません。オブジェクト指向方法論が普及していないように思われますが、どこに原因があるのでしょうか。

A. オブジェクト指向が普及しない原因のひとつとして用語の問題があると思われます。使用される用語がこれまでの開発方法論と違い、哲学的な言葉が使用されているのが特徴であると言えます。オブジェクト指向に取り込もうと　する前に、言葉の壁にぶつかり、あきらめる人が多いようです。また、今のところシステム開発に新しい手法を取り入れる必要性を実感していないことも原因で　あると考えられます。

何と言っても最大の原因は日本国内において、実例が非常に少ないことであり、オブジェクト指向による効果がどれほどのものであるかが、不明確なことにあるようです。

パーソナル・コンピュータなどハードウエアの低価格化が進むにつれ、企業や家庭における普及率が高くなってきています。それに伴い、ユーザーの要望も多　様化、その要望も短いサイクルで次の要望へと変化している現状を見てみると、これまでの開発方法論では、修正にかかる工数や規模が大きくなり、すぐに応えることはできません。まずは、オブジェクト指向に取り組む姿勢が重要であると思われます。

Q. オブジェクト指向を導入するには、どのようにすればよいのでしょうか？

A. オブジェクト指向はこれまでの機能指向の開発方法論とは逆の発想　を持ったものであると言っても過言ではないでしょう。それだけに、導入メンバーや実際に開発するメンバーは、この考え方に柔軟に対応できることが重要で　す。まずは、社内において少人数で小規模のシステムを開発してみてはどうでしょう。できれば、オブジェクト指向を学んできた人や実際に開発したことのある　人がそのメンバーに入っていれば理想的ですが、それも難しいと思います。自分たちでやってみて、自ら評価し、社内に普及させていけばよいのではないでしょ うか。

Q. エンドユーザーコンピューティング（EUC）において、オブジェクト指向は有効に作用するのでしょうか。

A. マイクロソフト社の Windows 等、GUI（グラフィック・ ユーザー・インタフェース）の分野では既にオブジェクト指向技術を使って開発されており、多くの人は知らず知らずのうちに、接してきているはずです。そう いった意味では、このオブジェクト指向はEUCに役立っているものと思われます。しかし、実際に開発するとなると、分析や設計はエンドユーザーにとって難 解なことも多いため、今のところ有効であるとは考えられません。社内やソフトウエア市場において、オブジェクトが蓄積されていくにつれて、簡単にそれを利用してエンドユーザーがシステムを構築する時代がやってくるかもしれません。その時には、オブジェクト指向の果たす役割が大きなものとなってくるでしょ う。

Q. オブジェクト指向へ移行した場合、これまでの資産はどのように扱われるのでしょうか。

A. オブジェクト指向自体は、ハードウエアを限定するものではありま せんから、これに対する影響は少ないと思われます。ただし、このハードウエアで稼働するソフトはオブジェクト指向で開発したそれと共存できるかどうかは疑 問です。既存のソフトをカプセル化することにより、共存可能とする意見もありますが、それを実施した場合、オブジェクト指向のメリットが半減する等意見が 様々です。いずれにしろ、この技術の導入を決定し、普及していけば、これまでのように自然に淘汰され、将来的には、この技術で開発されたものへと順次移行 されていくのではないでしょうか。

4．2 オブジェクト指向開発

Q. オブジェクト指向開発と従来の開発では、費用や工数・生産性にどのような効果が期待できるでしょうか。

A. オブジェクト指向開発で初めてシステムを開発する場合、手探りの 状態ですので工数及び費用、いわゆる初期コストはかかるでしょう。しかし、オブジェクト指向はシステム変更等に柔軟に対応ができるため、保守における費用 の削減は期待できるでしょう。生産性についても同じことが言えると思います。
　オブジェクト指向は共通化や標準化がしやすく、管理も容易であるため、システム規模が大きくなるにつれ、その効果は次第に向上していくものと考えられます。

Q. オブジェクト指向でシステムを開発する場合、C＋＋やSMALLTALK等オブジェクト指向言語を導入し、新たに教育を実施しなければならないのでしょうか。

A. オブジェクト指向に限らず、どのような方法論においても、それを 効果的に支援することのできる言語を導入し、開発することがベターです。これまで利用してきたプログラム言語（COBOL等）でオブジェクトを構築していきたいとの要望が多くなれば、供給されるようになるのでしょうが、開発言語については、開発事例が増えるにつれて、その在り方が議論されるようになるのか もしれません。

Q. オブジェクト指向開発において、設計者と開発者との意思の伝達はどのように行われていくのでしょうか。

A. これまでの開発方法論と同様に、設計書をもって意思の疎通を図っていくものとなります。しかし、これまでのものとは大きく異なり、よりビジュアルなものとなっていき、文章による記述は次第に少なくなっていくものと思われます。この点については開発サイドとエンドユーザーとの間も同じことが言えますので、これまでと違った設計書を提示することで、ユーザーからクレームの つくことのないよう、事前に教育あるいはオブジェクト指向の概要を説明しておくことが必要であると思います。

また記述方法についても、何種類かあり、標準化はされておりません。どの記述が良いのかは、導入する際に、事前に十分な検討が必要であると思います。自社にあった記述方法（方法論）を選択するとよいでしょう。

4．3 その他

Q. オブジェクト指向開発を担当させる要員には、どのような教育を実施すればよいのでしょうか。

A. オブジェクト指向は、未だに標準化が行われておらず、発展途上の 技術ではありますが、既にDB（データベース）や設計ツールなど世の中に出始めています。技術が確立されるのを待って、教育を考えるよりも、少しでも早い 時期に実際にシステムを構築するなど、経験を通して教育を実施することが良いのではないでしょうか。事例が少ないために、経験者を募っての教育は、困難に 思われます。

また、教育の成果を性急に求めることのないよう、長い目で見ることも大切なようです。これまでの開発方法論に固執しない、若年層を中心としたプロジェクトを作れば思ったより早く技術を確立できるかもしれません。

Q. オブジェクト指向方法論を支援するCASEツールの状況はどのようになっているのでしょうか。

A. オブジェクト指向CASEは主に米国で開発されたものが多く、 様々な分析手法や設計手法を支援しています。例えば、Schlaer & Mellor 手法のCASEツールとし て Teamwork/OOA、Coad & Yourdon 手法としては OOATool,OODTool、Booch 手法で は Rose 等があげられます。これらのツールは、上流工程から下流工程を完全に支援するものは少なく、高価なものです。機能的にも充実しており、 投資しやすい金額のCASEツールの出荷が待ち遠しいところです。今後の動向に注目しましょう。

日本国内においても、少しずつその種類も増えていくことでしょう。

第5章　新たなシステム構築の在り方

5．1　ユーザー要求を満たすシステム開発方法

「これからの時代に通用する新たなシステム構築の在り方とは？」という質問に一言で答えるとすれば、そ れは、ユーザー要求を百パーセント満足できるシステム開発方法ということができる。コンピュータの出現以来、今日までコンピュータシステムの構築、導入に より確かにユーザーは業務の効率化、即時化、自動化により多大な恩恵を受けてきたが、その構築の主導権は終始、コンピュータあるいはソフトウエア提供側に あったと言っても過言ではない。ユーザーの要求はコンピュータやソフトウエアの機能、処理能力が足りないからという理由で制約され、膨大な業務プログラム の開発に時間がかかるという理由で使い始めるまで長期間待たされ、さらに、競争相手が導入しているからといって投資効果に見合うかどうか確信のない高価な システム開発費、維持費を払わされていた。しかし、今日、パソコンが家電なみの価格で大量に販売され、世界的な規模で低価格化、大量販売競争が展開される等、コンピュータの利用に関する自由競争が当たり前になっている。したがって、情報システムに関する主導権、選択権は、すでにユーザー側に渡っており、 ユーザーの要求を満足できるコンピュータ／ソフトウエアベンダーだけが生き残り一人勝ちする時代が到来しているのである。このような背景から、システム開 発方法もユーザー満足度中心に変わらざるを得ない時期になっている。

ユーザー要求を満足するシステム開発方法の要点は、「速い」「良い」「安い」の3点であり、オブジェクト指向技術は、それを支え、実現するキーとなる技術ということができる。

本節では、はじめに、新たなシステム構築方法について、具体的にユーザーは何を要求し、どこまで満たせば満足できるのだろうかについて示し、次節で個々の要求とオブジェクト指向技術の関係を明らかにしたい。

（1）速い

文字どおり短期間でシステムが完成し、稼働することが最大の要求である。速ければ速いほど満足度は高い けれども、情報システムの場合、インスタントラーメンのように3分間というわけにもいかないので、新規開発の場合、仕様が固まってから3か月以内、注文を受けてから6か月以内で完成できれば十分に満足していただける範囲と考えられる。もう一つ、速いの中に、変更が速いという要求が含まれている。変更の場合 は、新規よりも一般にさらに速い応答が要求される。内容によっては、日単位の即応性が要求される場合もある。

（2）良い

操作が簡単で、使い勝手が良いこと。かゆいところに手が届くような機能設計と実装がされていること。高 い品質で故障がほとんどないこと。要求の変化に応じて自由に変更が可能なこと。これらがユーザーが要求する「良い」システムの条件である。もちろん、手間 と暇を

かければ限りなく良いシステムが実現できるはずなので、妥当な価格と開発期間の範囲であることが前提になる。

（3）安い

ユーザーが真に望んでいるのは、最近、流通サービスで行われている価格破壊をソフトウエアの開発にも起 こして欲しいということである。ソフトウエア産業は長い間、注文生産のシステム構築に馴染んできたため、コスト削減の努力を長く続けているものの、劇的な 価格の変化に対する取り組みが不十分である。

5．2 短期構築の方法とは

では、短期にシステムを構築するにはどのような方法が考えられるだろうか。原理的に3つの方法があり、新たなシステム構築では、現実的にはその組み合わせで開発期間を短縮することになる。

（1）プログラムを極力作らない方法

最も短期間でのシステム構築が可能な方法である。この方法には、

①出来合いのアプリケーション・パッケージソフトや部品の再利用によりプログラミングを最小にしてシステムを構築する手段
②土台となるシステムの上で、第4世代言語（4GL）やCASEを使ってユーザー自身でシステムを構築する（End-User Development：EUD）手段

の2種類が存在する。

①のアプリケーション・パッケージソフトを利用してシステムを構築する方法は、言い換えれば、大型のソフトウエア部品化とその再利用を行っていることになる。オブジェクト指向による大型部品化、再利用の技術を活用すれば、より短期の構築が期待できる。

（2）プログラムを細分化し並行開発する方法

大規模なソフトウエアシステムの開発においては、実際には、機能詳細化により適当な規模のモジュールに 細分しこれを並行開発し、順次組み上げていく方法（いわゆるウオーターフォール型開発）が一般的である。しかし、この方法は大規模になればなるほど細分化したモジュール間のインタフェースの調整、設計、確認のためのコミュニケーション稼働が指数関数的に増加し、多大な開発期間を必要とする結果を生ずる。
したがって、この方法による場合はインタフェースが疎で独立性の高いサブシステムへ、い

かに分割するかがキーの技術となる。データ中心設計とオブジェクト指向分析、設計法がこれを解決する技術として有望である。

（3）ユーザー要件を短期に正確に獲得する方法

短期の構築を困難にしているもう一つの問題は仕様の獲得、確定に時間がかかっていることにある。特にウ オータフォール型の開発では、ドキュメントベースによる仕様確認のため見逃された問題点が実際に使用が可能になる完成間際の段階で発覚し、ユーザーから問題が指摘されるため手戻りが多く生じることによる開発期間の遅延の例が後を絶たないのが実態である。

このような問題を解消し、ユーザー要件を短期に、しかも正確に獲得できる技術としてプロトタイピング技術が注目されている。プロトタイピング技術は特に ユーザーインタフェースに関する部分である画面インタフェース（GUI）に焦点をあてて行われるのが一般的であるが、短期間でGUIのプロトタイピングを 行い、ユーザーと対話的に仕様の確認を行うための有効なツールとして、オブジェクト指向で簡易にビジュアルプログラミングを行えるツールが市販され普及し ている。

5．3 使い勝手をよくする方法とは

（1）操作が簡単で、使いやすいシステム

第一に取り組まなければならない課題は、よりよいユーザーインタフェースを設計、開発することである。 そのためには、家を建て、外観、内装を設計するときにインテリアデザイナという専門家が対応するように、システム開発の専門家でGUIの設計をするのでは なく、ユーザーインタフェース専門のデザイナーという専門家を養成し、デザインのセンスを持ち込んだGUIの設計を行うことである。システムの開発がコー ディング・デバッグという力まかせの時代は昔の話であり、今や芸術的センスを有するデザイナーやプロデューサが必要な時代であることを認識しなければなら ない。

（2）高品質

オブジェクト指向でつくられたシステムは部品となるオブジェクトの動作、品質が追加や変更によって影響されないという意味で、従来型の開発よりも高品質であると言える。特に、6章で詳細に述べるビジネスオブジェクトを再利用できればなおさらである。

（3）要求に応じて自由に変更可能

オブジェクト指向技術は、次の2つの面でシステムの変更に対する柔軟性、即応性を可能とする。

1)ドラッグ＆ドロップによるカスタマイジング

オブジェクト指向で開発されたアプリケーションはユーザー画面上でのアプリケーションオブジェクトのドラッグ＆ドロップの操作で自由に、かつ簡単に組み合わせ、変更が可能となる（by Oliver Sims；Business Objects〔McGRAW-HILL〕）。

2)オブジェクトモデルの変更の容易性

オブジェクト分析・設計により開発されたシステムは、現実の世界の概念モデルがシームレスにシステ ムとして構築されているため、修正個所の特定が簡単で、かつ影響範囲が一目瞭然となる。したがって、フレームワーク、クラスライブラリの再利用、差分コー ディングで直ちに変更が可能であると同時に、長年のメンテナンスを経てもスパゲッティ化することは避けられる。

５．４ 価格破壊の方法とは

ソフトウエアの価格破壊を可能とするためにはソフトウエアの生産工程を徹底的に工業化することが必要で ある。工業化による価格破壊の原理は、大量生産（薄利多売、自動化、ロボット化）、人件費の削減（分業、海外生産）、組立工程の省略（大型部品化）、在庫 率削減（看板方式）等があるが、知的生産作業であるソフトウエアにこれらの原理がどのように当てはまるだろうか。

（1）大量生産

1）薄利多売

コピーが簡単にできるソフトウエア本来の特性から、薄利多売は最も本命の価格破壊の方法である。 パッケージソフトウエアOSについては、すでに値下げ競争が行われつつある。またリピータビリティの少ないソフトについては部品化、再利用によるリピータ ビリティの向上がキーであり、オブジェクト指向による部品化、再利用技術を普及できればさらに価格が下がることになる。

2）自動化

プログラムのコーディング、テスト段階での自動化はすでに普及しており、今後劇的な効果は期待できない。

（2）人件費の削減

1）分業

ソフトウエアの開発工程は、分析、設計、製造、保守に分けられ、最も単純で人手のかかる製造工程を 人件費の安い会社、国に分業することはすでに実施されている。したがって、工程による分業の効果は人件費の高騰による価格の上昇を押さえる効果はあって も、価格破壊につながるものではない。しかし、ソフトウエアの開発パラダイムをオブジェクト指向のパラダイムに完全に移行できれば、抜本的な価格破壊につながる可能性もある。すなわち、図5－4－1に 示すように、オブジェクト指向のモデリングにより、必要なクラスライブラリを設計、製造できるスーパ・プログラマと、このクラスライブラリを再利用、修正 する一般プログラマによる分業が可能になれば、グループ開発、分散開発や真の意味での分業がスムースにできることになる。

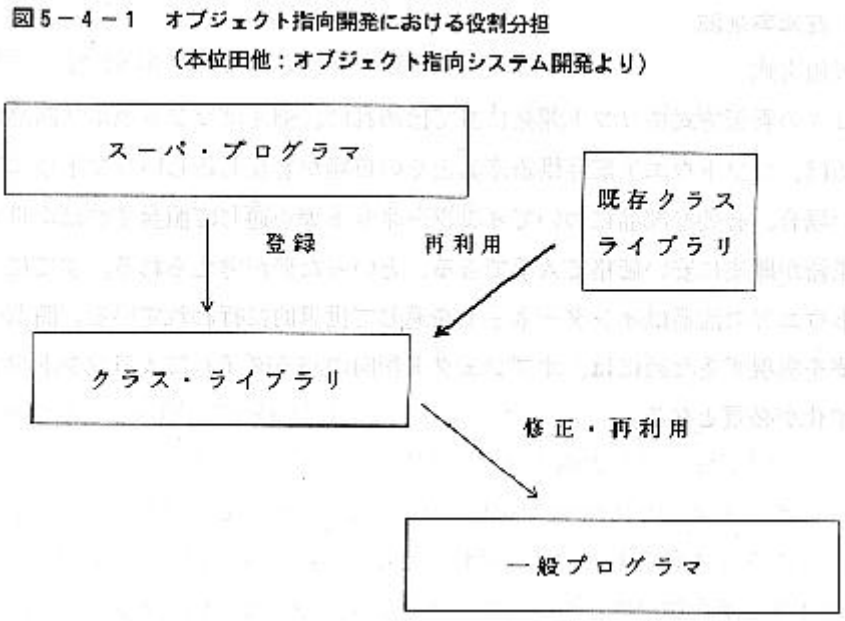


図5－4－1

2)海外生産

1)の分業の結果として、人件費が安く質の良いソフトウエアを生産できる国である、インド、中国等の国への生産拠点の移動がすでに始まっている。

(3)組立工程の省略

1)大型部品化

汎用性の高い、大型の部品を使って即座にユーザー要望を満たすソフトウエアを開発することで大幅な コスト削減が期待できる。大型の部品化技術としてオブジェクト指向技術の活用が必要であるが、特に、最近、注目されているビジネス・オブジェクトの考え方 はビジネスモデルそのもののフレームワークを95％以上再利用しシステムの構築を可能としようとする

もので、これが普及すれば、画期的な価格破壊が期待で きる(ビジネス・オブジェクトの概念は、6章で詳しく述べる)。

(4)在庫率削減

1)看板方式

トヨタの看板方式をソフト開発に当てはめれば、例えばソフトウエア部品の標準化が進めば、ソフトウ エア部品供給産業とその市場が普及し新しいソフトウエアを作成したい場合、必要な部品についてインターネット等を通じて照会すれば全世界から部品の供給が 即座に安い価格で入手できる、といった夢が考えられる。すでに、フリーソフトウエアの流通はインターネットを通じて世界的に行われている。商業ベースでこの夢を実現するためには、オブジェクト指向のパラダイムによるソフトウエア部品の標準化が必須となる。

第6章　ビジネス・オブジェクトー新しいオブジェクト・コンセプトの胎動

6．1　ビジネス・オブジェクトとは何か

オブジェクト指向技術に関係する人々の間で“ビジネス・オブジェクト”という言葉が注目を浴びてきている。いったいビジネス・オブジェクトとはなにものか？

この章では、その概要を紹介する。

最初に“ビジネス・オブジェクト”（以下、BOと略する）なる言葉が出てきたのは、OMGにBOMSIG（Business Object Management SIG）が出来上がった93年末である。BOMSIGが定義しているBOは図6－1－1のとおりであるが、簡単に言えば、人、場所、物、コンセプトなどをオブジェクトの形で表現したものといえる。では、なぜBOのコンセプトが出てきたのかは、BOMSIGチェアマンのロバート・シェルトン氏の書いたレポート（参照1）に述べられているように、ビジネス・プロセス・リエンジニアリング（BPR）が進む中で企業の経営者、ITのユーザーが一層激化する競争環境の中で生き 残っていくための商品開発、サービスの迅速な提供にはこれまでの発想を転換し、コア・ビジネス以外においてはビジネス・プロセスそのものまでも市場から調 達してきてもよいとの考え方が出てきているとの認識が原点にある。

図6－1－1

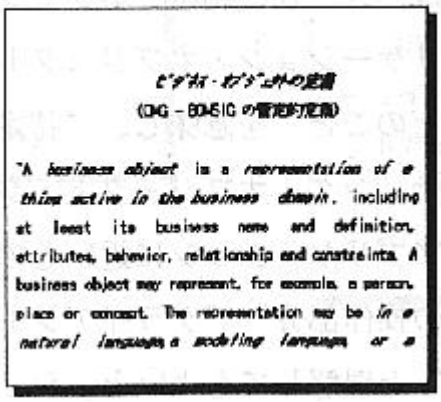


図6－1－1

言い換えれば、企業のビジネス・モデルを共有し、再利用しようという考え方が出てきているのである。つまり、BOの考え方は、オブジェクト指向技術をビジネス・モデルの作成・変更やビジネス・システムの構築に活用しようという発想である。

これまでのオブジェクト指向プログラミングでいうオブジェクトとは、図6－1－2の 中でテクノロジー・オブジェクトに該当しており、OSから始まりファイル、入出力、ディスプレイ処理等の物理的機能を受け持つオブジェクトや、テキスト処 理・グラフィック・マルチメディア・分散コンピューティングのためのオブジェクト（ディストリビューション・サービス、ディレクトリ・サービス、メッセー ジング・サービス等）などのクラス・ライブラリまでを意味していると理解すればよいと考える（参照2）。

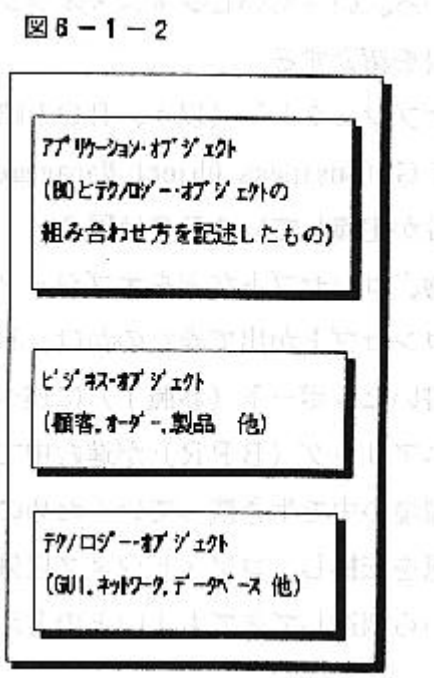


図6－1－2

ビジネス・オブジェクトは“人、物、場所、コンセプトなどをオブジェクトの形で表したもの”であり、ア プリケーション・オブジェクトは“注文入力などといった個別業務アプリケーションなどのこと”を意味し、“特定のアプリケーション・ドメインをカバーする コーポレーティング・オブジェクト・クラスの集まり”であり、“ほとんど完結した、すぐ動くアプリケーションに近いもの”といわれている。また別の言い方 をすれば、“GUIの操作部分”“クライアント・ソフトあるいはプレゼンテーション機能をもつソフト”と理解してもよいと、シェルトン氏は言う（参照3、4、5）。ロバート・シェルトン氏が定義する各々のオブジェクトの説明は付録に添付する。

6．2 OMGにおけるビジネス・オブジェクト関連の動き

先ほど、OMGのBOMSIGのことに触れたが、OMGそのものもユーザーサイドからのアプローチが必要であることを十分認識しており、94 年中からは業種ごとのインダストリー・バーティカルなSIGが次々とできつつある（参照2）。

これらのSIGの活動内容は今後非常に注目されているが、最近、BOMSIGの活動計画の中でビジネ ス・オブジェクトをサポートする製品を提供しているベンダー、またビジネス・オブジェクト・コンセプトを導入・設計しているユーザーのサーベイが実施されることが決定しており、この結果が 95 年第3四半期頃には出版されるので興味深い(参照6)。

OMGのSIGそのものは、テクニカル・コミッティーと異なり標準そのものを策定するものではないが、 最近の動きによれば、金融業のSIGはコモン・ファシリティ(CORBAファシリティに改称)のTCに対しRFPをリクエストしたとの話であり、いよいよ OMGの中でビジネス・システムそのものをサポートするビジネス・オブジェクトのための機能が検討されるタイミングが来ていると考えられる(参照6)(図6－2－1)。

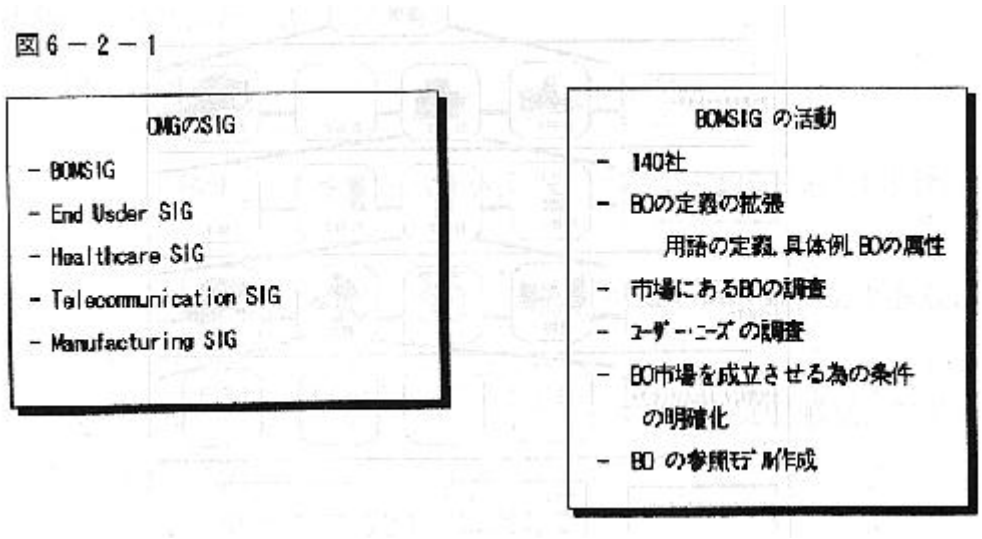


図6－2－1

6．3 情報システムの中におけるビジネス・オブジェクトの位置づけ

これまでのプログラミング中心のオブジェクト指向技術で考えているオブジェクトをプログラミング・オブジェクト(またはテクノロジー・オブジェクト)として定義するとき、ビジネス・オブジェクトとの関係を表したものが、図6－3－1である。まだ明確に各々を整理したものは筆者の知る限りないと考えるが、現在、一つの考え方として情報システムをレイヤーの中で区分けするときそれぞれ位置づけるコンセプトが出てきているので紹介する(参照7)。

図6－3－1

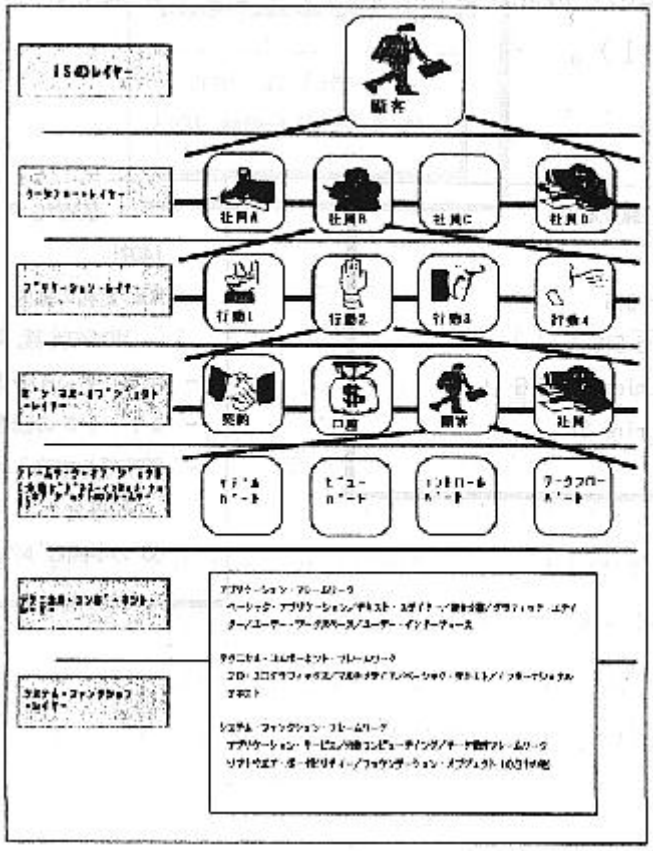


図6－3－1

まず第1番目がエンドユーザー・コンピューティングのレイヤーで、ここでは、あるユーザー（社員Aとす る）が顧客に情報を製品として提供しながらPC画面上のオブジェクト（アイコンと同様な画面上の対象物）をドラッグ・ドロップしながら情報の抽出／ナビ ゲートをしているレイヤーである。

第2番目が、あるビジネス・プロセスの各タスクを複数の社員が各々の役割を果たしながら規定されたフ ローの流れに従って実行しているワークフローのレイヤーである。ここでは、社員は自分が遂行するべき一連の仕事（フローに従って個々のアプリケーション・ オブジェクトが構成された形で社員のワークプレース上に展開されている）を自分のワークプレースを使って行っている。

第3番目のレイヤーは、2番目のレイヤーでいう個々のアプリケーションの“集合”したレイヤーである。 ここでは、ワークプレースを作るための“ワークプレース・コンストラクト（土台）”と選択されたオブジェクトを使い“特定のアプリケーション”を作っているレイヤーである。

第4番目のレイヤーは、アプリケーションで扱うべき対象物（顧客、物、場所、コンセプトなど）となるオ ブジェクト（ビジネス・オブジェクト）を作り上げているレイヤーである。ここでは、オブジェクト・フレームワークなる器を使いアトリビュート／アクション ／アノテーションなどを仕様に従いダイナミックにバインドし“顧客オブジェクト”“場所オブジェクト”“地図オブジェクト”などのオブジェクトの集まりを 作り上げている。

第5番目のレイヤーは、フレームワーク（枠組み）が集まったレイヤーであり、ここにはビジネス・オブジェクトを作り上げるため、またアプリケーションを構成するための機能・ツール群が集まっている。

例としては（参照8）、

－ワークスペース・フレームワーク

①個人、ワークグループ、組織、役割、エージェント等の“人”に関わるプログラマブル・デスクトップ
②オフィス、会議場所、部門、企業、顧客、ベンダープロジェクトなどの“場所”に関わるデスクトップ
③プリンタ、ファックス、電話、入出力ボックス、ツール、備品、ブリーフケース等の“物”に関わるデスクトップ

－グラフィック・エディター
－テキスト・エディター
－複合文書フレームワーク（OpenDOc 等）
－データ変換ツール
－ユーザー・インタフェース・ツール

などである。

第6番目のレイヤーは、テクニカルなコムポーネントが集まったレイヤーで、ここには、

テキスト・レイアウト／国際テキストの変換／2D・3Dグラフィックの表示・印刷／マルチメディア

等の機能を提供するフレームワークが集まっている。

第7番目のレイヤーは、システム・ファンクションの集まりであり、ここには、

－アプリケーションを構築するためのサービス機能

テスト／フォント／カラー／ビデオ装置の選択／時間管理／データアクセス等

－分散コンピューティングの機能

メール・メッセージング／システム管理／ディレクトリサービス／分散オブジェクトのサポート（DCE、CORBAなど）

－オブジェクト・サービス・インタフェース機能

ランタイム／マイクロカーネル／OO型デバイス・ドライバ／ファイルシステム、ネットワーキングなどOSの各種機能

などである。

それでは、このレイヤーの中で特にビジネス・オブジェクトに注目すると、このレイヤーの下のレイヤーは 各ソフト・ベンダーの固有なソフト環境の中でいろいろなフレームワークが提供されることが予想されるが（つまり個々のベンダーの間のオブジェクトの互換性 が問題になるが）、ここから上のレイヤーでは、ビジネス・オブジェクトをどう組み合わせてアプリケーションの流れを定義し、フローとしてつなげていくか は、ビジネス・プロセスの設計そのものであり、一般に言う情報システム部門の外の問題領域とみなせる。つまりユーザー部門の責任で行われる範囲が広がることになり、これまでの情報システムがユーザー部門に開放されることになり、新しいプロセス設計の世界が開ける可能性を秘めている。

別の言い方をすれば、情報システム部門は5番目のレイヤーまでを使い、4番目のレイヤーのビジネス・オ ブジェクトを設計し、第3番目のレイヤーであるアプリケーション（ここで言っているアプリケーションとは分散環境で話題となる Unit-of -Work″またはタスクと理解する方がわかりやすいと考える）構築の基礎までを設計する。アプリケーションとはユーザーがワークフローのレイヤーまで定義して初めて完成する、またはユーザーがPC画面上のオブジェクトにアクションをとって（ドラッグ・ドロップ）して初めて流れとして完結するという意味である。この世界はこれまでのアプリケーションの概念を一新し、これまでの“Application-Centric-Computing″から “Task-Centric-Computing″ の世界へ変わると言われていることに符合する（参照9）。

6．4 オブジェクトを使ったソフトウエア・ファクトリーの考え方

新しいオブジェクトを基本としたソフトウエア生産の考え方（ソフトウエア・ファクトリーと呼ぶ）が、どういう流れで生産されていくかについて少し詳しく考察する。

図6－4－1の ように、情報システムの開発をレイヤーで考える。6つのレイヤーは、その他のレイヤーの部品の上にのっかって作られる。ソリューション・フレームワークは テクニカル・コンポーネント・レイヤーのオブジェクトを再利用する。ビジネス・ドメイン・レイヤーはソリュー

ション・フレームワーク・レイヤーのオブジェ クトを再利用する。同様に、上位レイヤーは下位レイヤーのオブジェクトを再利用する関係になる。

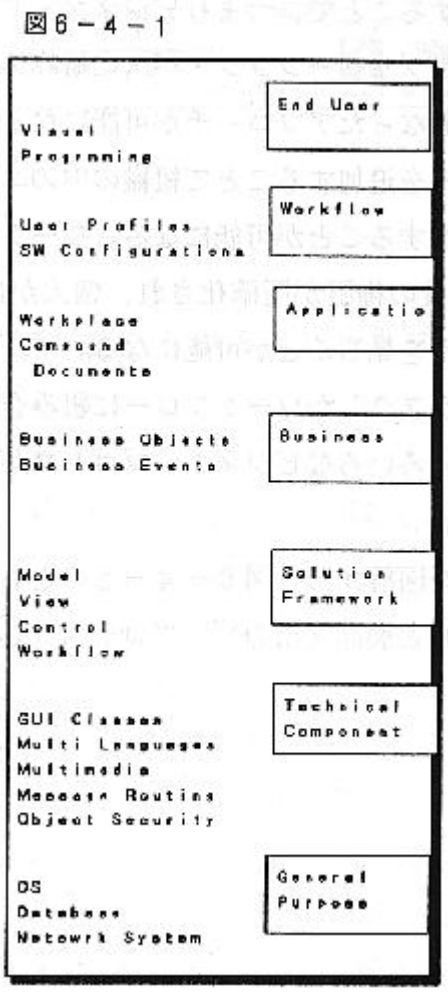


図6－4－1

依然として一般的なレイヤーである。つまりビジネスに固有な情報はまだ組み込まれていない。この上の4つのレイヤーは、企業の内部のIS部門が生産する責任を持つことになるレイヤーである（インターナル・ソフトウエア・ファクトリー）。

ビジネス・ドメイン・レイヤーを追加することは、生産のスピードを飛躍的に向上させることになる。ただ し、メタ・ソリューション・レイヤーでのオブジェクトの再利用というのは同様なネットワーク・アーキテクチャを持つビジネスの中においてのみ実現可能であ ることに十分注意することが必要である。

アプリケーション・レイヤーを追加することは、ビジネス・ドメイン・オブジェクトの組み合わせ方（アプリケーション）を決めることであり、これまでの伝統的な開発活動はここのレベルで完結する。

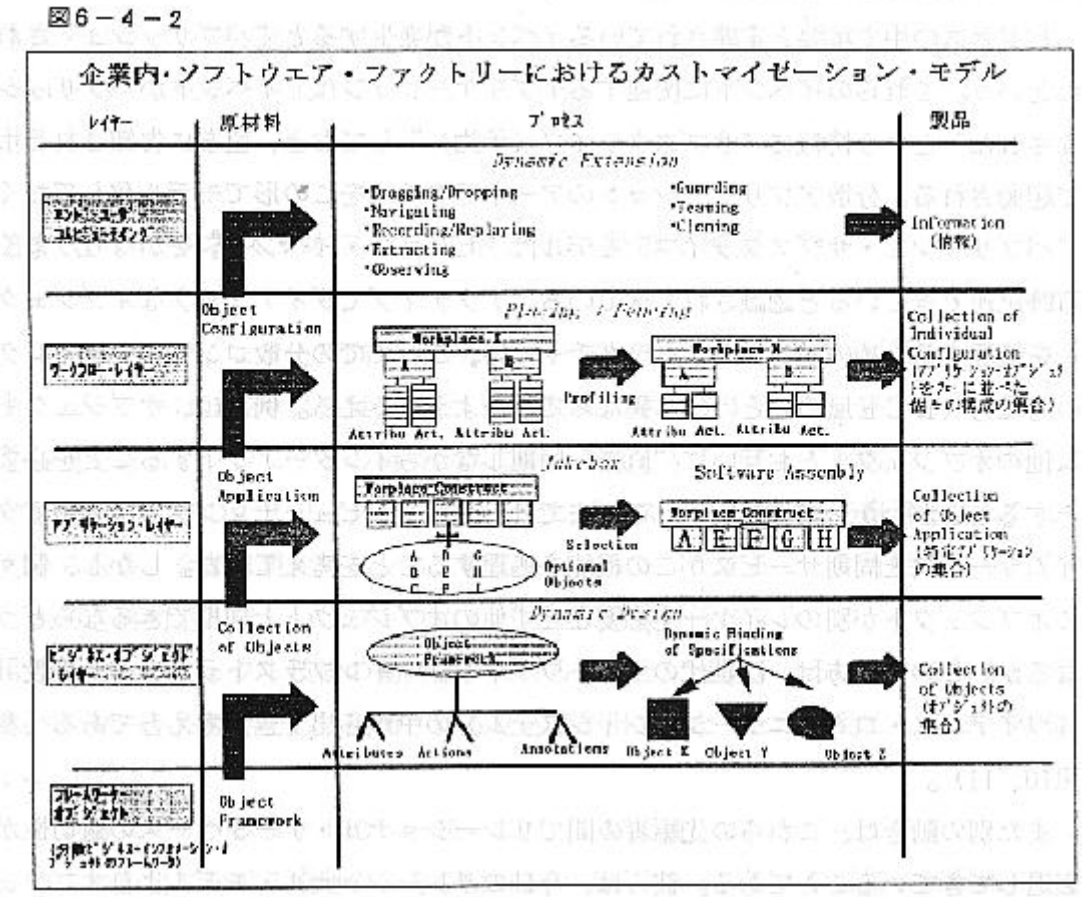
ワークプレースを提供することで、つまりビジネス・ドメイン・オブジェクトをユーザーがダイナミックにアプリケーション・パスに組み込めるようなコンティナーを作ることで、これまでと異なったアプローチが可能になる。

ワークフロー・レイヤーを追加することで組織の中のユーザー部門にビジネス・プロセスの設計の権限を委譲することが可能になる。ワークフロー・レイヤーを使うことで、各部門の中での社員の権限が明確化され、個人が自分に完全に“カスタマイズ”されたISの ありようを見ることが可能になる。

また、個々の社員の“タスク”がワークフローに組み合わされることで、“ウォールからウォール”までのいろいろなビジネス・プロセスが、このレイヤーを組み入れることで作り上げられる。

以上、述べてきたことを図解すると図6－4－2のとおりである。この図により各レイヤーの原材料（入力）と製品（出力）、プロセス（処理）の関係が明確になる。

図6－4－2



6．5 ビジネス・オブジェクトから“ビジネス・イベント”の考え方へ

最近開催されたパトリシア・セイボルド・グループのパイオニア・ミーティングに参加した分散オブジェク ト・コンピューティングの最先端のリーダーたちの関心事は、これまでの静的な、オブジェクト指向型のビジネスのスナップショットとしてのオブジェクトでは なく、“動的オブジェクト”に変わってきていることだと言われている。彼らは、ビジネス・オブジェクトの考え方を理解し使っているが、今、“ビジネス・イ ベント”という観点から、考え始めている。言い換えれば、一番面白いビジネス・オブジェクトは“ビジネス・イベント”、つまり“受注されたオーダー”“出荷された製品”“処理されたカストマー・プロブレム”等であるという(参照 10)。“ビジネス・オブジェクト”と“ビジネス・イベント”の関係はビジネス・オブジェクトがいろいろな“ステート(状態)”を遷移する中の一時点を意味するものと理解されるが、いまだ完全な理解には至っていない。

ビジネスの中で重要と定義されているイベントが発生すると“パブリッシュ”されたという。これらのイベ ントに関連するアプリケーションは“イベントがパブリッシュされた”という情報に“サブスクライブ(予約)”しておき、直ちに告知される形で起動される。 分散アプリケーションのアーキテクチャをこの形でモデル化しておく“パブリッシュ・サブスクライブ”モデルは“ビジネス・イベント”をかなりうまく追跡記 述できていると認識されてきている。アクティブでダイナミックなオブジェクトを処理するためのインフラストラクチャーは、これまでの分散コンピューティン グの考え方(DCE風の)とは少し異なっているようにみえる。例えば、オブジェクトは他のオブジェクトとお互いに“直接”同期しながらインターアクトする ことを必要とすることがわかってきている。これまでは、分散コンピューティング環境の中でタイムサービスと同期サービスがこの機能を処理することを考えて きた。しかし、個々のオブジェクトが別のレイヤーを必要とせず他のオブジェクトと同期できるならどうなるか。この考え方は、次世代のネットワーキング・イ ンフラストラクチャーの設計(ワイアレス・コミュニケーション・システム)の中から出てきた考え方である(参照 10、11)。

また別の動きは、これらの先駆者の間でリレーショナル・データベースの適切性が衰退してきていることで ある。彼らは、今日のリレーショナル・モデルよりオブジェクト・データベースまたはハイブリッドデータベースによって、よりうまくサポートされるダイナ ミック・オブジェクト、“イベント・トリガード・アクション”(イベントが引き金となってアクションが起動される)機能を持つ“リアルタイム・アーキテク チャ”を、より好ましく思っていることである。これからのリアルタイムで、ダイナミックな、分散情報処理の世界に向かうにつれ、購買パターンや顧客のビ ヘービヤーの変化を、それが起こるつどとらえていく能力の方が単なるビジネスのスナップショットをとることより、より面白く、かつまた重要になる。

ミドルウエアが分散コンピューティングの中でこれまで以上に重要な役割を果たすことになるが、 “Lean”で“強固”な分散コンピューティング・システムを構築していくためには、個々のトランザクションのオーバーヘッドを小さくしていかなければいけないので、ミドルウエアも“薄い”ものが重要になる。クライアントはアクティブで、サーバーはサービスが要求されるまではパッシブのままでいることがで きるようなクライアント/サーバーシステムが必要になる。

ここで最も重要なサービスとなるのは、とりわけセキュリティ・サービスとディレクトリー・サービスである。ビジネス・プロセスを記述するワークフローはビジネスそのものを設計するデザイナーと、それを支える技術を実装していく人たちの間で、より重要 な要素としての位置を占めてくるようになる。

6．6 ビジネス・オブジェクトのこれからの展望

6年前にオブジェクト指向ツールがコマーシャル・ユースに利用可能として登場以来、ユーザーはビジネス の世界において“現実の世界（リアルワールド）”の考え方を表現できるオブジェクトの構築の話に焦らされてきている。“顧客”“オーダー”“販売区域”等 のオブジェクトを利用してビジネス・コンセプトを構造化することができるなら、企業は現在のビジネス戦略をシミュレートすることが可能になり、また、これ らを再利用し新しいソフトウエアが迅速に構築可能になる。

最近のオブジェクト技術の動向は、これらのことが可能であるばかりでなく、少数のベンダーから製品として出荷される見込みが出てきたことから見ても非常に興味深い。

ビジネス・オブジェクトのユーザーにとっての価値、また、そのビジネス・オブジェクトの市場における流通の仕方を検討したビジネス・モデルの詳細（参照 12） がレポートされているので、ぜひ読まれることをお薦めする。また4月 25 日～28 日にマサチューセッツで開かれるパトリシア・シーボルド・グループの “1995 ディストリビューテッド・オブジェクト・テクノロジー・フォーラム”では、詳細な議論とユーザー事例が紹介される予定なので、ぜひ参加されるこ とをお薦めする。加えて、オブジェクト指向技術をわかりやすく解説させたら右に出る者はいないと言われているDavid Taylorがつい最近新著を出しており、ビジネス・プロセス・リエンジニアリングとビジネス・オブジェクトの関連を解説しているので、合わせて読まれれば全容が理解できるものと思う。

以上、断片的な知識、情報の直訳を羅列してきたが、ビジネス・オブジェクトの考え方はこれまで情報シス テムに携わってきた方々にとって新しい視点を切り開くものであり、BPRの嵐の中で本来の情報システムの目的を追求する手段として必須のものであると信じ て紹介させていただいた次第である。

〔参照文献〕

1．“The Distributed Enterprise″, Robert E. Shelton, Distributed Computing Monitor 10-93 Patricia Seybold Group
2．“オブジェクト指向ビジネス・エンジニアリングとビジネス・オブジェクトの展望” Robert E. Shelton, OSPG 秋季フォーラム講演集
3．“Co-operative Business Objects″, Oliver Sims McGraw Hill IBM Series,1994

4. 日経データプロ速報版 1994.11
5. 日経データプロ速報版 1994.12
6. OMG BOMSIG Meeting Minutes, Feb-95 TC Meeting
7. “You can bank on objects″, Robert Prins, Cyclade Consultants レポート
8. Taligent Application Environment, IBM Object Technologies Brochure
9. “The Future of Object Technology″ James A. Cannavino, Object Magazine Nov.-Dec 1994
10. SnapShots, Feb 1995, Patricia Seybold Group
11. Patricia Seybold Group’s “1995 Distributed Object Technology Forum″ -Brochure
12. “Business Objects″, Distributed Computing Monitor, Jan 1995, Patricia Seybold Group
13. Business Reengineering and Business Objects, David Taylor

【付録】　ロバート・シェルトン氏の提唱するオブジェクトの定義

ビジネス・オブジェクト

・ A Modeling or software package of business procedure,policy & controls around data about a single concept
・ A way to organize the right data and right procedure in the right place
・ Independent of applications – used across the enterprise to represent shared business concept like customer, order or product

テクノロジー・オブジェクト

・ A Modeling or software package used in the information technology infrastructure,representing a single technology concept
・ A way to organize technology in a standard architecture based on off-the-shelf and custom parts
・ Used to build business obiects, applications and system services

アプリケーション・オブジェクト

・ Custom-Built or purchase business tools that perform specific business tasks or present information
・ Assembled from program code and techlology obiects.
・ Clients of business object services.
・ Presentation layer services, such as order entry, customer maintenance, billing, claims processing.