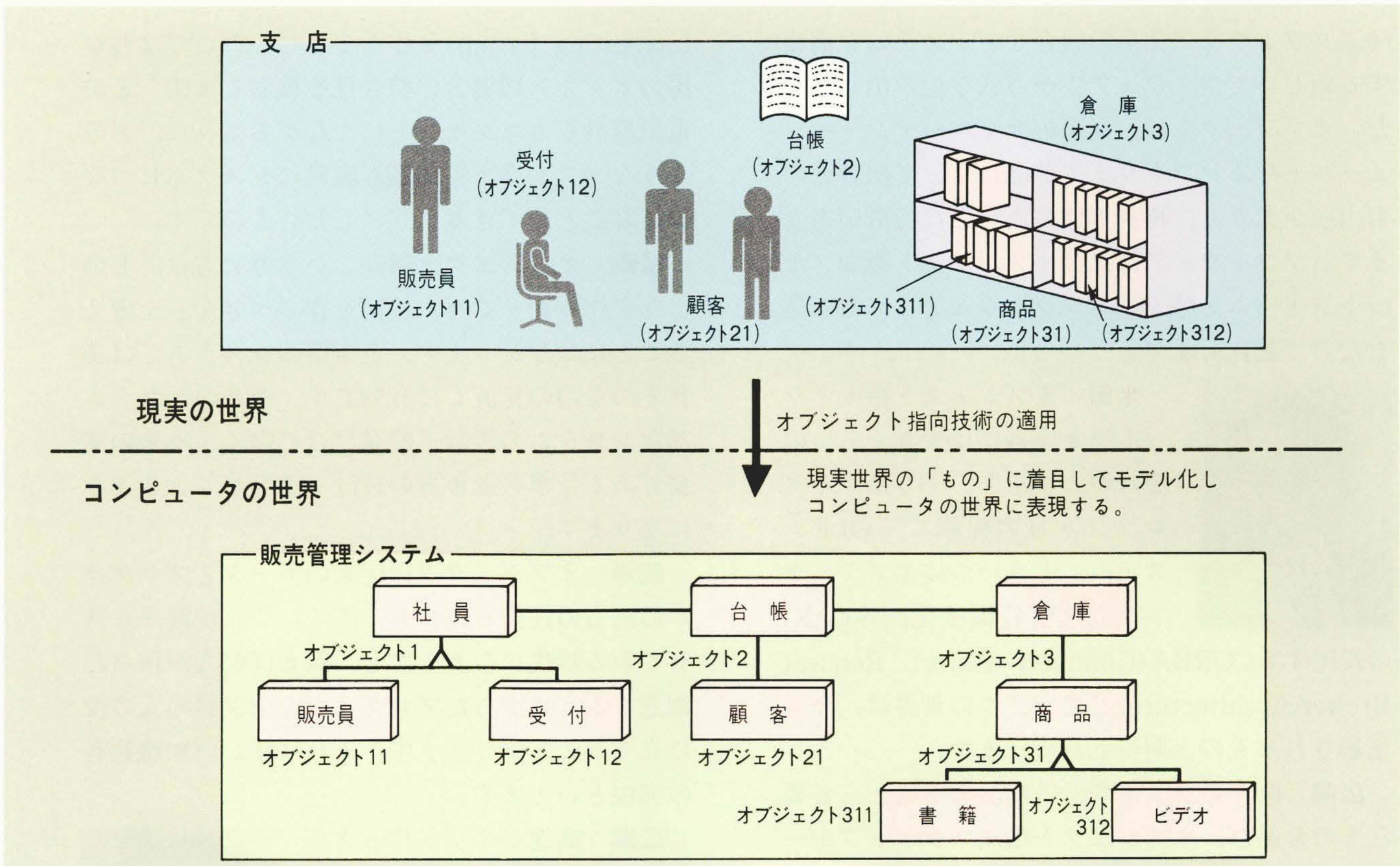
オブジェクト指向技術の動向

Trends of Object-Oriented Technology

岸本芳典*　*Yoshinori Kishimoto*　齋藤眞人***　*Masato Saitô*
広瀬　正**　*Tadashi Hirose*　梶山雄三****　*Yûzô Kajiyama*



オブジェクト指向によるシステム構築

オブジェクト指向では，現実の世界をそのままコンピュータの世界に写し込むようにしてシステムを構成する。そのため，高い拡張性，保守性，再利用性が実現でき，システムの開発・拡張に要する期間・コストを低減する。

情報システムを活用する企業では，ビジネス環境の変化に頻繁に対応できるシステムを低コストで高効率に変更し拡張できる方法を求めている。

この要求にこたえて登場したオブジェクト指向技術は，現実の世界に存在する「もの」に対応したソフトウェア部品を作り，その組み合わせでシステムを構築する新しいソフトウェア構成法である。そのため，仕事や書類を「もの」としてとらえれば，システムの変更部分はこの変更部分に対応したソフトウェアの「部品」を修正するだけで済むことになる。

このたび開発したシステム構築技法は，オブジェクト指向技術に基づいて，企業情報システムの変更や拡張を容易に構築するものである。また同時に，オブジェクト指向開発支援ツール群を製品化した。これらのツールは，ユーザー自身がシステムを開発する際に，その効率向上に利用できるものである。

一方，大規模な企業情報システムもオブジェクト指向を用いて生産効率よく構築・運用できるように，オブジェクト指向データベース管理システムなど，業界標準に合致した各種のミドルウェアも製品化した。

* 日立製作所 システム開発研究所　** 日立製作所 情報事業本部　*** 日立製作所 ソフトウェア開発本部
**** 日立製作所 情報システム事業部

1　はじめに

'90年代のビジネス環境はその変化が激しく，かつ競争はますます激化してきている。このような環境では，企業にとって製品・サービスを早く市場に出すことが成功のかぎとなってきている。そのため，各企業のビジネスを支える企業情報システムや産業システムに求められる機能は急激・急速に多様化・高度化し，現行システム以上の柔軟さ，開発期間・費用の削減，投資対効果の向上が要求される。これを実現するためには，ソフトウェアの機能変更・追加，およびシステム能力・規模の拡張が容易であることが必要になる。前者に対しては部品化・再利用によるソフトウェア構築が有効であり，後者に対してはワークステーション・パソコン(パーソナルコンピュータ)をLANやWAN(Wide Area Network)による分散システム構成にしてスケーラビリティを確保することが有効である。この両者を実現する技術としてオブジェクト指向が注目されている。

オブジェクト指向とは，現実世界の「もの」に着目することを意味し，ソフトウェアを直観的でわかりやすく，再利用や変更を容易にする。これらの特徴によってオブジェクト指向技術は，従来の構造化技法に代わる，新たなソフトウェア開発技法として期待されている。また，大規模な異機種分散環境でのシステム構築では，相互運用性や移植性，再利用性が重要な課題であり，オブジェクト指向技術はこれを実現する中核技術としても有効である。

ここでは，オブジェクト指向技術の動向と，日立製作所のオブジェクト指向への取組みについて述べる。

2　オブジェクト指向とは

2.1　オブジェクト指向の特徴

オブジェクト指向とは，実在物(例：商品，台帳)や概念物(例：支店，受付)などの現実世界に存在する「もの」に着目してモデル化を行い，コンピュータ上で表現する考え方である。オブジェクト指向技術の説明には抽象化，カプセル化，継承ということばが使われ，それぞれの意味と効果は以下のようにまとめられる。

(1)　抽 象 化

「もの」に共通な性質を整理して「クラス」とする。例えば，現実世界の個々の本やビデオテープに共通の性質を「書籍」や「ビデオ」というクラスにまとめる(**図1**参照)。個々の本やビデオテープは，クラスに属する「オブジェクト」として表現される。これにより，自然な発想でソフトウェア化でき，ソフトウェアの理解が容易となる。

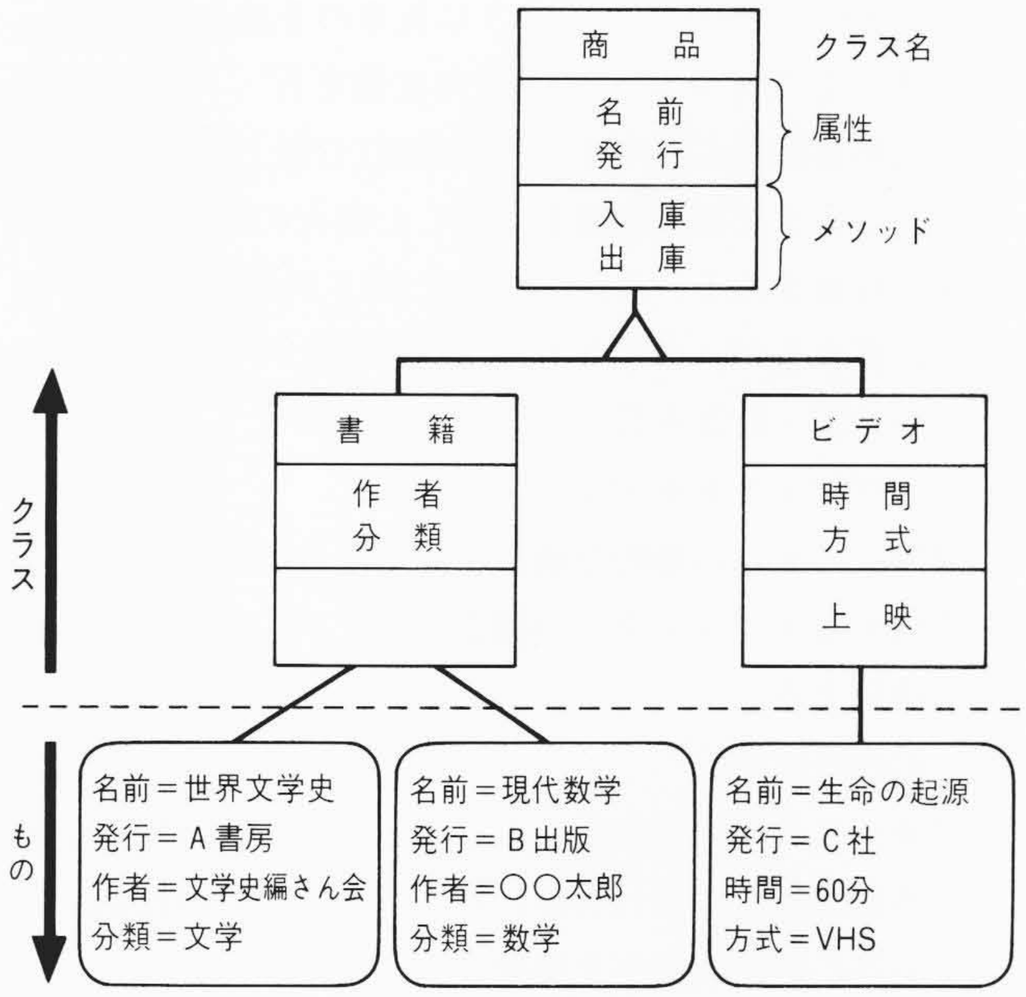


図1　オブジェクト指向によるモデル化
現実世界の「もの」に着目し，ものに共通の性質をクラスとして抽象化する。さらに，クラス間に共通の性質を上位のクラスとすることによってクラス階層が構成される。

(2)　カプセル化

オブジェクトはデータを格納する「属性」と，データを操作する手続きである「メソッド」で構成する。外部(他のオブジェクト)から属性を参照・変更するためにはメソッドを呼び出さなければならない。このカプセル化により，あるオブジェクトの変更が他のオブジェクトへ与える影響が少なくなるため，信頼性が向上する。

(3)　継　　承

クラス間の共通の性質を上位クラスとしてまとめるとクラスの階層ができる。例えば**図1**に示すように，「書籍」や「ビデオ」に共通の性質を「商品」という上位クラスにまとめる。下位のクラスはその上位クラスの性質(属性，メソッド)を自動的に引き継ぐ。これを継承という。既存のクラスを拡張・変更するには，継承機構を用いて既存のクラスの下位クラスとして拡張・変更部分だけを追加すればよいので，変更が容易になる。

2.2　オブジェクト指向開発技法

オブジェクト指向技術は元来プログラミング技法の一つとして考案された。そのため，まず各種のオブジェクト指向プログラミング言語が開発されてきた。これには，Smalltalk，Eiffelのように新たに開発された言語や，

C++, OOCOBOLなどのように従来の手続き型言語を，ベースとしてオブジェクト指向拡張を行った言語がある。近年は開発工程のより上流段階(分析・設計段階)からオブジェクト指向技術を適用した種々の分析・設計方法論が提案されている。手順や記法は方法論によってそれぞれ異なるが，いずれも基本的には次の3種類の側面からの分析・設計を行うものである。

(1) オブジェクトモデル

対象システムの静的な構造を表すモデルで，システムを構成するオブジェクトの構成とオブジェクト間の関係を明確化する。

(2) 動的モデル

対象システムの動的なふるまいを表すモデルで，システム外部からの刺激(入力)に対するシステムの反応，および応答の時間的順序を明確化する。

(3) 機能モデル

対象システムの内部的な計算動作を表すモデルで，ふるまいに必要な入力から出力への一連のデータ変換の流れを明確化する。

これら三つのモデルは，それぞれ相補的な関係にあるので，あるモデルを決めることによって他のモデルを再度見直すことが必要になる。システム要求がかなり明確な場合には，オブジェクトモデルから始めるのが効果的である。一方，システム要求が必ずしも明確でない場合には，動的モデルから始めるのが効果的である。また，機能モデルは省略されることがよくある。

分析・設計段階からオブジェクト指向技術を適用することにより，分析からプログラミングまでオブジェクトという一つの統一した観点で作業でき，現実問題として避けられない「手戻り」に強くなる。

2.3 分散オブジェクト環境

異なったメーカーの異なったマシンがネットワークで接続された分散環境(異機種分散環境)では，各計算機上で稼動するソフトウェアが連携して処理を行う必要がある。したがって，機種やプラットフォーム，通信方式，プログラミング言語などの差異を隠ぺいして相互運用性を持つことが非常に重要である。また，異機種分散環境上のシステム開発を容易にし，分散環境でのソフトウェア機能およびシステム構成の拡張・変更に対応するためには，ソフトウェアの移植性，再利用性が重要である。これらを実現するためには，それぞれの計算機上にあるソフトウェア部品をオブジェクトとみなし，すべてのソフトウェア部品があたかも1台のマシン上にあるかのように扱えるようにできるとよい。このように分散システムのニーズに対してオブジェクト指向技術を適用したマルチベンダ・マルチプラットフォーム分散環境を「分散オブジェクト環境」という。

分散オブジェクト環境上のオブジェクトは，通常のオブジェクトと同様にデータと手続きを一体化(カプセル化)したもので，他のオブジェクトのメソッドを呼び出す(メッセージを送る)ことによって連携する。メッセージを送る際に，送り先のオブジェクトが分散環境上のどこに位置するか，メッセージ伝達のための通信方法は何かなどの分散システムの構成や使用するプラットフォームに依存した処理は，分散オブジェクト環境のメッセージ伝達機構が行う。これによってアプリケーションの開発者は，ソフトウェア部品のネットワーク上の位置や実装の詳細を知らなくても利用できるようになる。また分散システムを意識せずに開発されたソフトウェア部品が，分散システム上でもそのまま使えるようになる。

2.4 オブジェクト指向技術の効果

タイムリーにかつ低コストでシステムを開発するかぎは，既存ソフトウェアの再利用である。再利用の形態には，既存のものに変更を加える場合と，既存のものをそのまま使う場合の二つがある。

既存のものに変更を加える場合には，(1) 理解容易性：既存のものが使えるか，どこを変えればよいかなどが簡単にわかること，(2) 変更容易性：変更を施す手間が最小限であること，(3) 独立性：ある変更が他の部分に影響を与えないことが必要である。一方，既存のものをそのまま使う場合には，(1) 理解容易性：既存のものが使えるかどうかが簡単にわかること，(2) 接続容易性：既存のものを部品として簡単に組み込めることが必要である。さらにいずれの場合でも，既存のものが再利用に値する品質でなければならない。

オブジェクト指向技術は，これらの再利用に必要な諸条件を備えている。理解容易性については，現実世界の「もの」を基本にしているため，出来上がったソフトウェアは直観的で理解しやすい。変更容易性については，継承機構を用いて変更による違いの部分だけを追加すればよく，既存のものを変えないで変更することができる。独立性については，カプセル化によってオブジェクトの内部の変更が他のオブジェクトに波及することがなく，かつ継承機構によって変更前のものと変更後のものとが共存できるため，変更による影響がきわめて少ない。接続容易性については，オブジェクト間のインタフェース

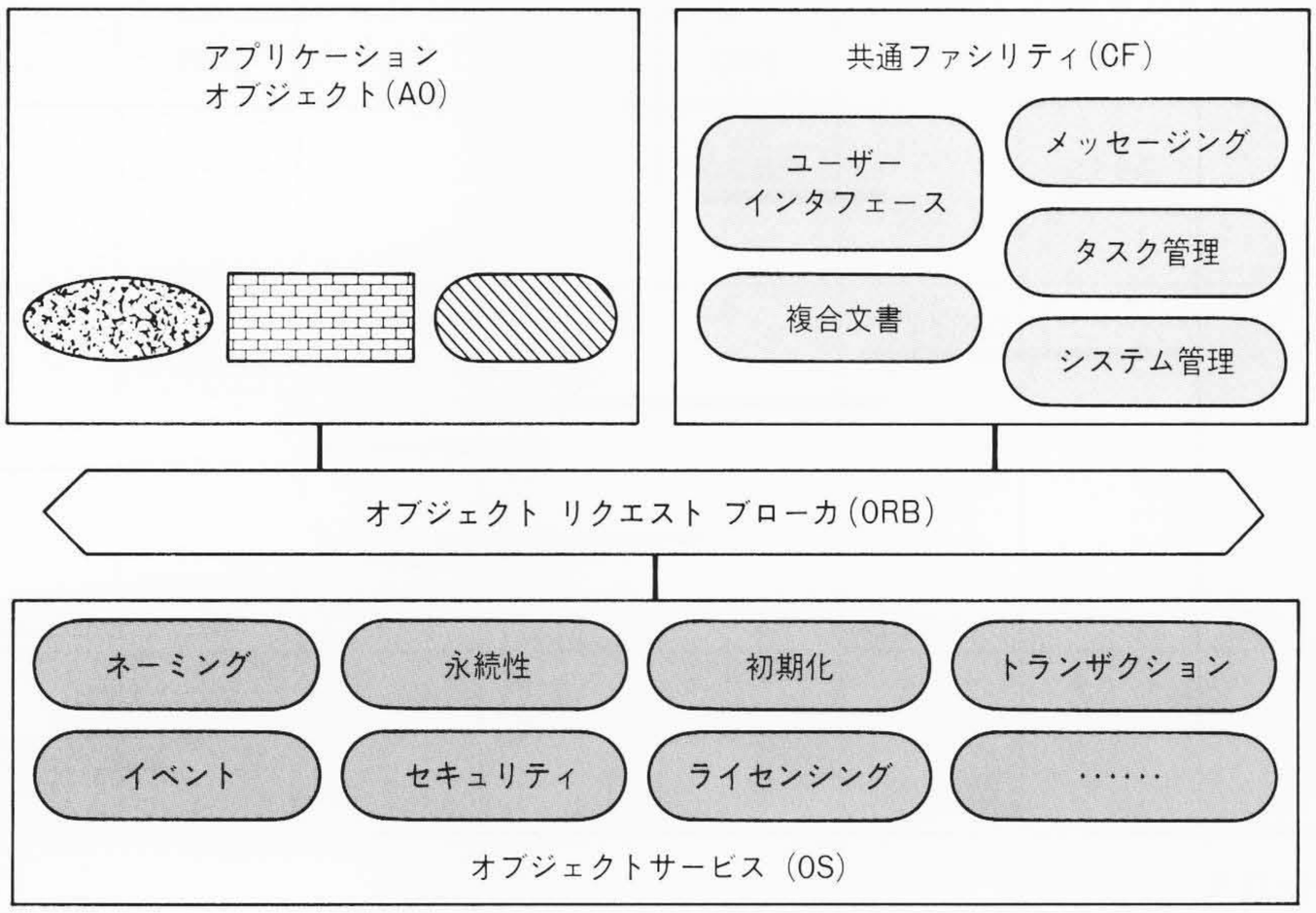


図2　OMGのレファレンスモデル
　レファレンスモデルは，標準化すべきオブジェクト技術の枠組みを提供するものである。オブジェクト リクエスト ブローカ，オブジェクトサービス，共通ファシリティ，およびアプリケーションオブジェクトの四つの部分で構成する。

がメソッド呼び出しだけに単純化されているため，接続についての制約が少ない。さらに重要な点は，この単純化は集中型か分散型かといったシステムのハードウェア構成に依存しないため，ハードウェア構成の変更による影響を受けないことである。

一方品質に関しては，ソフトウェア構造とその理解容易性・独立性のため，誤りの混入が未然に防止でき，高品質なソフトウェアが得られる。

このようにオブジェクト指向技術は，ソフトウェアの開発・拡張に要する期間・コストの低減と，分散システム化によるシステムの拡張性向上・コスト低減化をもたらす。

3　オブジェクト指向技術の動向

3.1　オブジェクト指向開発技法の動向

オブジェクト指向言語の歴史は古く，'60年代のSIMULAにまでさかのぼる。一躍有名になったのは'80年代に登場したSmalltalk80による。ただし，Smalltalkそのものは実行性能や実行環境の制約から当時はあまり広くは使われなかった。C＋＋の登場により，オブジェクト指向言語がなじみあるものとなった。その後のウィンドウシステムを備えたワークステーション・パソコンの普及に伴ってGUI(Graphical User Interface)を用いたプログラムが一般的となり，'90年代になってGUIオブジェクト(ボタン，ボックスなど)に処理を付加する形式のビジュアルなプログラミング言語VB(Visual Basic)やVC＋＋(Visual C＋＋)が登場した。これらのビジュアル言語により，GUI中心で比較的小規模(分析・設計をあまり必要としない規模)のオブジェクト指向プログラミングは容易になってきた。

一方，オブジェクト指向分析・設計技法としては現在までに多数の方法論が提案されている[1)]。現時点で最も利用者が多いのは，OMT(Object Modeling Technique)とブーチ法である。OMTは，図表やモデルが詳細に決まっており，かつ分析・設計の作業手順が具体的に示されている点が評価されており，適用例も多数報告されている[2)]。OMTは動的モデルの記述が不十分であるとの評価があったが，最近になって他の技法の記法を採り入れるなど動的モデルの記述の拡張・強化が行われている。一方，ブーチ法は図の種類が豊富であり，必要な図はほとんど網羅しているが，それらの使い方・手順が明確には規定されておらず，技法の利用者が個別に用意する必要がある。

3.2　分散オブジェクト環境の動向

分散オブジェクト環境はさまざまな手法で実現できるが，基本的にマルチベンダ・マルチプラットフォームを目指したものであるので，標準仕様が必要となる。分散オブジェクト環境の標準化，および製品の代表的なものとして，OMG(Object Management Group)による標準仕様の策定[3)]，Microsoft社のOLE[※1)](Object Linking

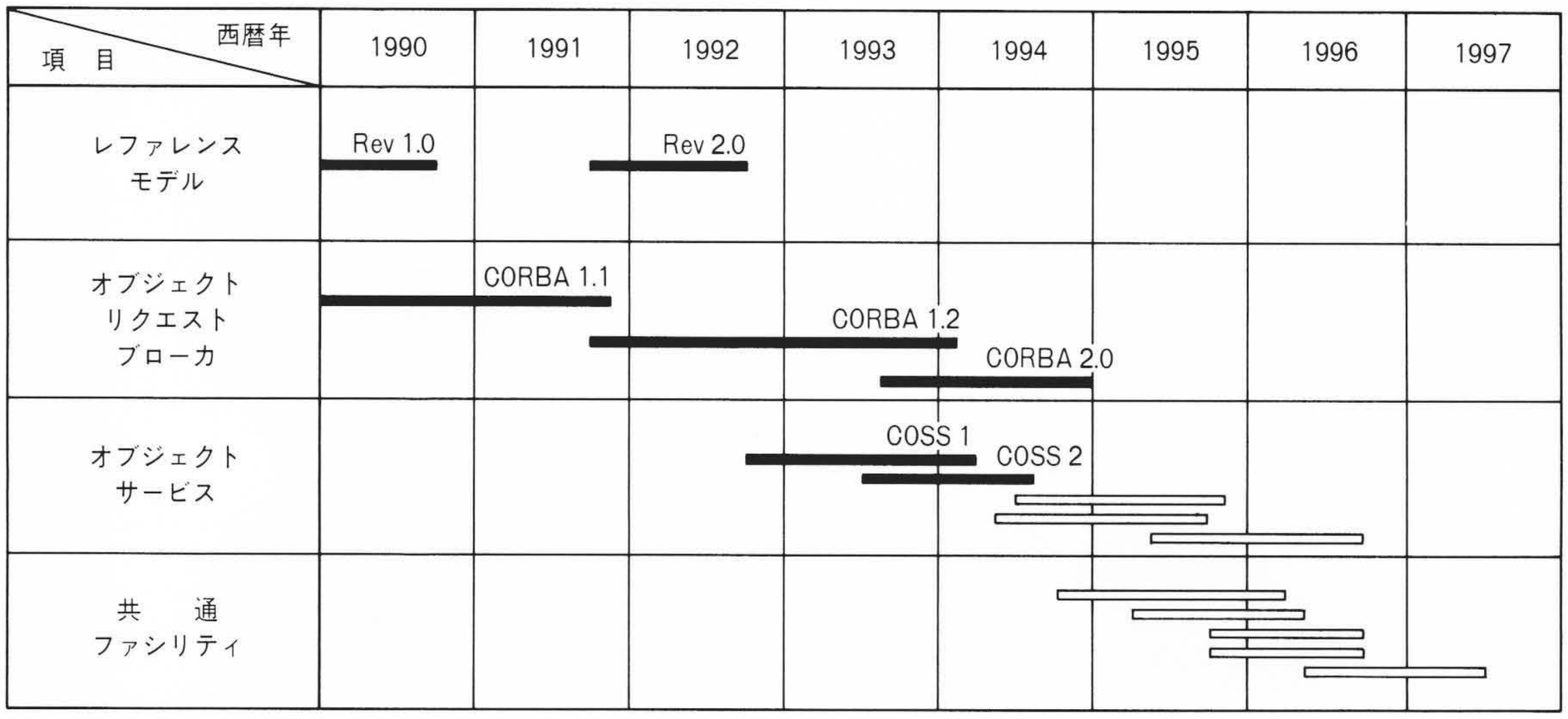


図3 OMGの活動経過と今後の予定
オブジェクトリクエストブローカは，CORBA2.0で異種ORB間連携の問題を解決した。オブジェクトサービスは，一部標準化が完了した。共通ファシリティは現在標準化作業が進行中である。

and Embedding)がある。

OMGは，オブジェクト指向技術の標準化に関する非営利のコンソシアムで，1989年に設立され，1995年1月時点で470社が参加している。OMGでは，分散オブジェクト環境のモデルであるレファレンスモデル(**図2**参照)を定め，これに沿って具体的な仕様の提案をメンバーから募り，審議・採択するというプロセスによって標準仕様を決めている。OMGの標準化対象は，オブジェクト間のメッセージ配送機構であるORB(Object Request Broker)，分散システムに必須(す)の基本機能を提供するオブジェクトサービス，アプリケーションに有益な機能を提供する共通ファシリティである。現在までにORBの仕様CORBA(Common Object Request Broker Architecture)とオブジェクトサービスの仕様COSS(Common Object Service Specification)の一部とが標準化され，共通ファシリティの標準化はこれから行われようとしている(**図3**参照)。

OLEは，Microsoft社が提唱するオブジェクトモデルCOM(Component Object Model)に基づくアプリケーション間の連携モデルである。複数のアプリケーションで作成したデータを組み合わせて新しい文書を作成する機能などを提供する。現在のOLE(OLE2)はまだ分散環境に対応していないが，分散環境への対応も計画されている。

OpenDoc[※2)]は，IBM社が提唱するオブジェクトモデルSOM(System Object Model)に基づくアプリケーション間の連携機構である。SOMはCORBAに準拠している。OpenDocはOLEと同様の機能を持つほかに，OLEに対応したアプリケーションの利用ができる。

現状ではOLEとCORBAとは互換性がないが，DEC社(Digital Equipment Corp.)とMicrosoft社は，DEC社のCORBA準拠のORBとOLEとの連携を可能にする，新たなオブジェクトモデルCOM(Common Object Model)を共同でOMGに提案している。

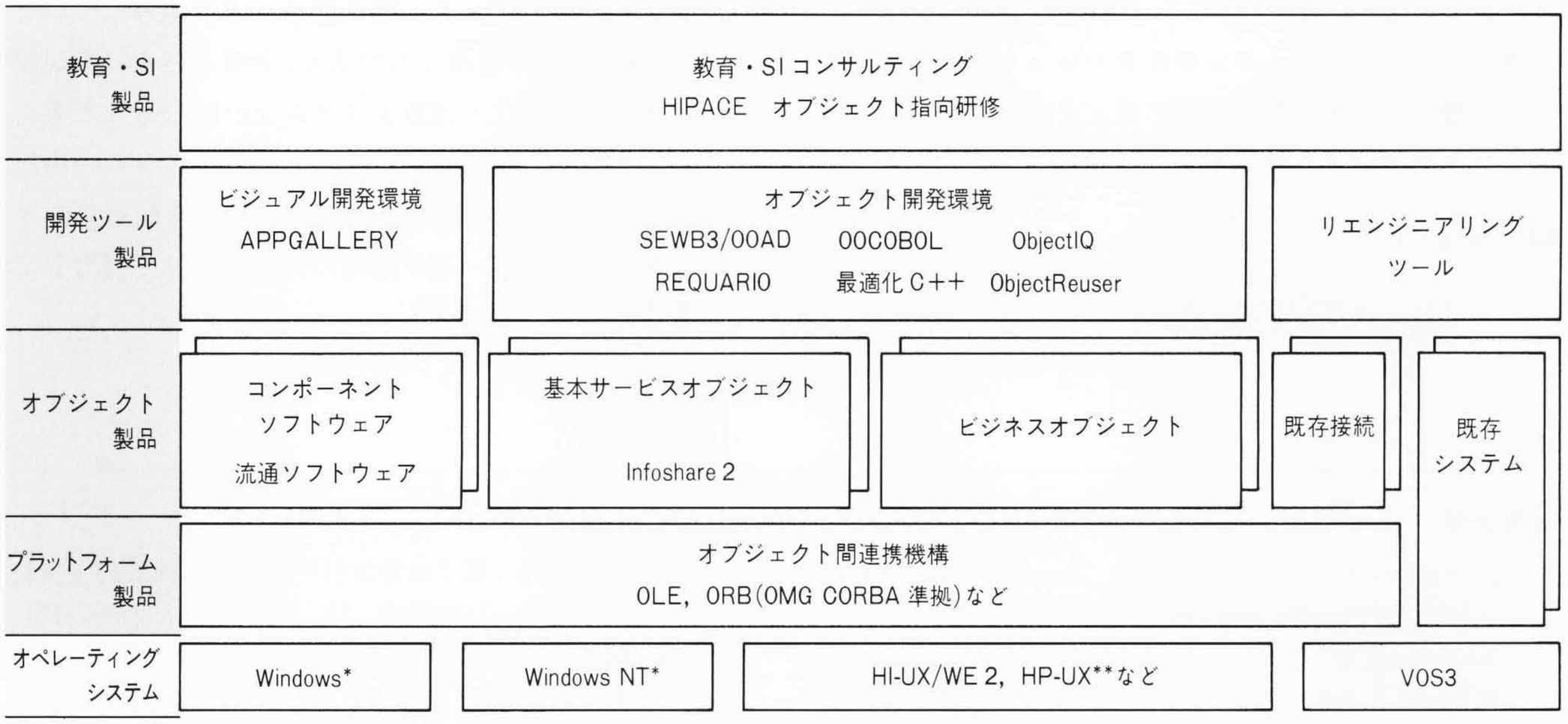
3.3 オブジェクト指向技術の適用状況

幾つかの分野ではオブジェクト指向技術の適用が試みられている。当初はオブジェクト指向技術の実験的試用による評価を目的としたものが中心であったが，最近では実用システム開発の前段階としての適用も増えつつある。よく使われるものは，オブジェクト指向分析・設計技法とその支援ツール(エディタ)，オブジェクト指向言語，アプリケーションには依存しないデータ型レベルの部品ライブラリまたはGUIの部品ライブラリ(ボタン，スケールなど)，OLEを利用するコンポーネントウェアなどである。OMGで定まった仕様に準拠した製品は，基盤となるORB製品が各社からようやく出そろってきたところであり，本格的普及はこれからである。

前述のような比較的小さな部品のライブラリは，汎(は

※1) OLEは，米国Microsoft Corp.が開発したソフトウェア名称である。

※2) OpenDocは，米国Apple Computer Inc.の商標である。



注：略語説明ほか　SI (System Integration)，HIPACE (Hitachi High-Pace)，REQUARIO (Requirement Scenario Designer)
SEWB 3/OOAD (Software Engineering Workbench 3/Object-Oriented Analysis and Design)
*　Windows，Windows NT は，米国 Microsoft Corp. の商標である。
**　HP-UX は，米国 Hewlett-Packard Company のオペレーティングシステムの名称である。

図4　オブジェクト指向ソフトウェア製品体系
オブジェクト指向システムの開発・運用に必要な四つの製品分野のソフトウェア製品を提供する。

ん)用的である反面,再利用による効果はおのずと限られたものになる。再利用性をより高めるものとして，アプリケーション業務そのものをクラスライブラリ化したフレームワークと呼ばれる業務パッケージが登場しつつある。フレームワークを使えば，素材となる部品を使って組み立てるのではなく，アプリケーションの設計結果そのものを再利用することができるため，生産性の飛躍的向上が期待できる。

4　オブジェクト指向製品

ワークステーションとパソコンの普及に伴い，CSS (Client Server System)がさまざまな分野で用いられはじめている。今後は，より大規模で論理的・物理的に分散したシステムが必要になると考えられている。

日立製作所は，オブジェクト指向製品を四つの製品分野に分けて開発し，生産性・柔軟性の高いオブジェクト指向開発・運用環境を実現する(**図4**参照)。

(1)　分散プラットフォーム製品

分散システムを一つの仮想的なシステムに見せる製品を示す。ここではオープン性(世界標準インタフェースへの対応)が重要であり，OMGのCORBAとMicrosoft社のOLEに準拠した製品を提供していく。

(2)　オブジェクト製品

アプリケーションの構成要素・基盤となる製品を示す。基本サービスオブジェクトには，トランザクション管理[4]，ワークフロー管理(Groupmax)[5]，複合文書管理，データベース(Infoshare2)[6]などがあり，整備が進んでいる。一方，ビジネスオブジェクトには，受注管理，在庫管理などが考えられているが,具体的製品はまだ少ない。今後，実験システムの構築経験を基に，本格的システムの構築を通じてこれを整備していく[7)~11)]。

(3)　開発ツール製品

オブジェクト製品を開発するためのツール群を示す。OLE部品の組立によるビジュアル開発環境(APPGALLERY)[12]，オブジェクト指向分析－設計－プログラミングツール(SEWB3/OOAD[13]，ObjectIQ，REQUARIO[14]，オブジェクト指向COBOL言語をベースとした開発環境[15]など)，再利用支援ツール(ObjectReuser)，既存ソフトウェア資産をオブジェクト指向環境で活用するためのリエンジニアリングツール[16]，既存ホストシステムとの連携を実現するラッピングツールなどを整備している。

(4)　教育・SIコンサルティング製品

業務改革，教育，システム設計を支援する製品を示す。オブジェクト指向言語・データベースの入門から実習まで，およびオブジェクト指向分析・設計演習などを行う

オブジェクト指向研修サービス,標準開発手順(HIPACE)[17]の導入を支援する導入コンサルテーションサービス，顧客の実態に合わせた開発手順の策定を支援する生産技術コンサルテーションサービスなどがある。

5 おわりに

ここでは，オブジェクト指向技術の動向，および日立製作所のオブジェクト指向製品への取組みについて述べた。現在CSSが普及しつつあり,今後はさらに大規模なシステムのCSS化・分散システム化が進むものと考えられる。今後も，オープン環境での分散システムの中核技術となるCORBA仕様などの標準化動向を配慮し，オブジェクト指向開発・運用環境の実現に向けて努力していく考えである。

参考文献

1) Monarchi, et al.：A Research Typology for Object-Oriented Analysis and Design, Comm. ACM, Vol.35, No.9(1992-9)
2) 情報処理学会誌：特集　オブジェクト指向分析・設計，情報処理，Vol.35，No.5(1994-5)
3) 大野，外：オブジェクト・マネジメント・グループとその活動，情報処理，Vol.35，No.9(1994-9)
4) 花塚，外：分散オブジェクト指向型トランザクション管理システム，日立評論，**77**，12，867～870(平7-12)
5) 藤崎，外：統合型グループウェアパッケージ－Groupmax－，日立評論，**77**，5，361～366(平7-5)
6) 和歌山，外：オブジェクト指向データベース“Infoshare2”，日立評論，**77**，12，863～866(平7-12)
7) 柳，外：西日本鉄道株式会社における乗務員ダイヤ作成システム，日立評論，**77**，12，875～878(平7-12)
8) 西川，外：データ項目部品を適用した人事システム—東村山市役所—，日立評論，**77**，12，879～882(平7-12)
9) 松本，外：北里大学東病院における薬歴管理システム，日立評論，**77**，12，883～886(平7-12)
10) 末光，外：東京証券取引所における株式約定接続支援システム，日立評論，**77**，12，887～890(平7-12)
11) 温井，外：営業支援地図情報システム“HMAP/ET”，日立評論，**77**，12，871～874(平7-12)
12) 冨永，外：部品組立型業務アプリケーション開発ツール“APPGALLERY”，日立評論，**77**，12，859～862(平7-12)
13) 高舘，外：オブジェクト指向によるシステム分析・設計支援ツール“SEWB3/OOAD”，日立評論，**77**，12，847～850(平7-12)
14) 斎藤，外：オブジェクト指向による要求仕様書視覚化ツール“REQUARIO”，日立評論，**77**，12，843～846(平7-12)
15) 西尾，外：プログラミング環境を充実させたオブジェクト指向COBOL，日立評論，**77**，12，855～858(平7-12)
16) 秋庭，外：既存システムの再構築を支援するリエンジニアリングツール，日立評論，**77**，12，851～854(平7-12)
17) 千葉，外：オブジェクト指向技術を用いたソフトウェアの開発技法，日立評論，**77**，12，839～842(平7-12)