UDC 681.3:332

金融情報系システム分野におけるIT

Information Technology for Financial Distributed Systems

岡村邦彦*(1) 山口 徹*(1)
Kunihiko OKAMURA Toru YAMAGUCHI

抄 録

高度化，多様化，グローバル化が進む金融取引業務を支えているのは，情報系システムそして情報技術(Information Technology)であり，その現状を概観した。クライアント・サーバ・アーキテクチャに基づくシステムがその中心となるが，金融アプリケーションの開発においてはオブジェクト指向技術とアプリケーション・フレームワークがシステムの拡張性・保守性を実現する上で特に有効である。

Abstract

With the growing diversification and technologinal advancement of today's business', distributed systems based on state of the art Information Technology(IT)are essential for their success, especially in the financial market. Client server architecture is held at the core of such distributed systems. It seems that the extendability and maintenance of financial systems are dramatically improved by the use of object-oriented technology and application frameworks.

1. 緒 言

日本版金融ビッグバンを控え，金融取引の自由化の波が押し寄せている日本の金融機関にとって，先端的な金融商品のディーリングからリスク管理まで，その競争力を支えているのは最先端のコンピュータを駆使したシステム技術，通信技術であるといっても過言ではない。本稿では，金融情報系システムを支えるIT(Information Technology)に関して，中心となるクライアント・サーバ・アーキテクチャとその周辺技術，更には最新動向であるオブジェクト指向技術とアプリケーション・フレームワークについて事例を挙げながら述べる。

2. 金融情報系システムとそれを取り巻く環境

本稿で対象とする金融機関の情報系システムの概要を図1に示す。市況情報配信システムは，ロイターなど外部情報機関より取得した金利，為替，株価などの市況情報をフロントシステムにリアルタイムで配信し，また，値洗いや分析計算用にデータベースに保存する。フロントシステムは，金利，為替，債券，商品やそれらの派生商品のディーリング業務を支援する。ミドルシステムでは，保有する取引のさまざまなポートフォリオに対して，市場リスクや信用リスクの計測と管理，そしてポジション管理を行う。バックオフィスシステムではメインフレーム系が主体であるが，勘定系と統合した次世代ALM(資産，負債の総合管理)は情報系に移行する動きもある。

このような金融情報系システムを取り巻く環境は大きく変化しつつある(図2参照)。今日まで多様な金融派生商品が出現している

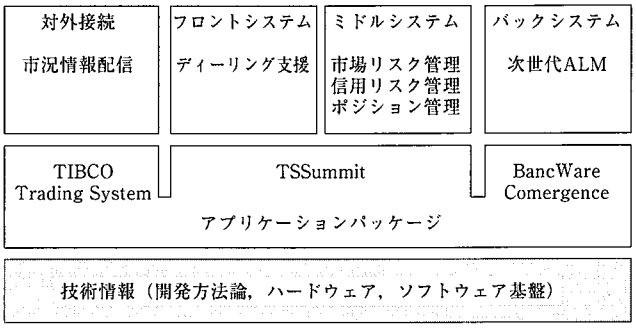

図1 金融情報系システムの構成

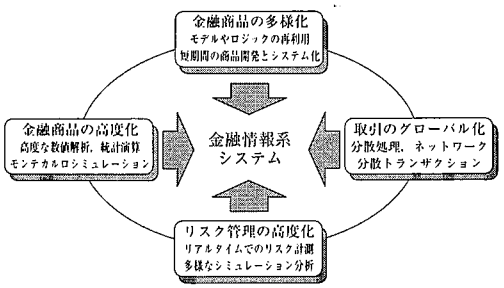

図2 金融情報系システムに求められる主な機能

*(1) エレクトロニクス・情報通信事業部
金融システムソリューション部 グループリーダー
東京都渋谷区代々木3-25-3 〒151-8527 ☎(03)5352-2206

が，特に商品寿命の短い証券分野では，商品開発の効率アップと短期間でのタイムリーなシステム化が求められている。またノックアウトオプションのように原資産価格の履歴に価格が依存する高度な金融派生商品も開発され，プライシングにあたっては偏微分方程式の高度な数値解析やモンテカルロ・シミュレーションなど多大な計算量が必要とされている。柔軟なモデリングと高度な分析・計算能力を提供しなければならない。更に海外拠点と連携したグローバル取引情報の収集，日中管理ベースでのポジション計測など，分散環境での高速な応答性が求められている。

次節では，金融情報系システムに対するこれらの高度な要求への回答として，オブジェクト指向技術をベースとするクライアント・サーバ・システムの技術を紹介する。

3. 金融情報系システム分野におけるIT

3.1 分析設計方法論

金融業務アプリケーションの分析設計においてまず注目すべき点は，分析対象となる金融派生商品のもつ自由度の高い発展性である。**表1**に主な金融派生商品を挙げるが，これら以外にも原資産となる取引に対してオプション取引条件や先渡し取引条件などを組み合わせることにより多様な新商品を定義することができる。そのためシステムのもつ機能を中心にトップダウンで分析を行い，それぞれの階層の機能を順次詳細化して設計を進めて行く従来の構造化設計手法では，このような無限の組み合わせをもつモデルの分析には自ずと限界がある。

このようなモデルに対してはオブジェクト指向分析設計法が有効である。**図3**に金融商品のオブジェクトモデル例を示す。オブジェクト指向分析設計では，どのような金融派生商品でも抽象的な金融商品クラスを継承して実現できる。たとえばオプション取引の原資産自体も抽象金融商品としてモデル化すれば，任意の組み合わせのオプションを記述することが可能である。

さらに金融商品の現在価値(Net Present Value)を計算するメソッドgetNPV()として抽象クラスで定義し，継承により具象化されたクラスとそのメソッドを実現する。そうすると多様な金融商品取引からなるポートフォリオの現在価値を計算するアプリケーションでは，取引ごとの個別の商品を意識することなく，すべて抽象的な金融商品としてgetNPV()メソッドを起動することで適切な現在価値計算メソッドが実行される。このように金融商品クラスライブラリとアプリケーションを構築しておけば，将来新商品が追加された場合でも，その新商品のクラス定義を加えるだけでよく，アプリケーションのコードにはまったく手を入れずにシステムの機能を拡張することができる。

表1 主な金融派生商品

対象	原資産	派生商品	
		取引所取引	相対取引
金利・為替	LIBOR, Swap Rate 長期プライムレート 短期プライムレート CD 為替レート	金利先物 金利先物オプション	金利先渡し取引 金利スワップ 金利オプション 為替予約
債券	国債 利付き金融債 社債	債券先物 債券先物オプション	アセットスワップ 債券現物オプション
株式	現物株式 株価指数	株価指数先物 株価指数オプション	エクイティスワップ エクイティオプション
商品	現物商品 （金，石油，ほか）	商品先物	コモディティスワップ コモディティオプション

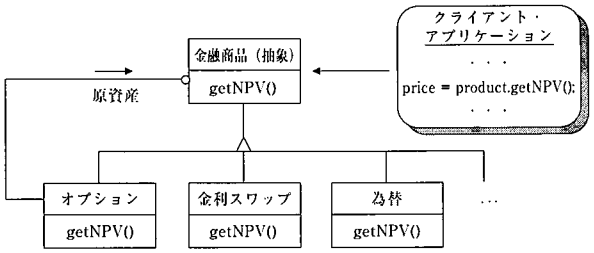

図3 金融商品のオブジェクトモデル

3.2 システムアーキテクチャ

情報系システムの主流をなすアーキテクチャは，クライアント・サーバ・アーキテクチャである(**図4**参照)。システムはデータベース・サーバとクライアント・アプリケーションの2層に分割されている。クライアント・サーバ・アーキテクチャは，中央集中型のアーキテクチャに比べて，モジュール化が容易で，機能拡張性に富み，かつ分散処理により信頼性や処理能力が向上している。

しかしながらシステムが大規模化してきた場合，2層のアーキテクチャではいくつかの問題点が指摘されるようになってきた。すなわち，クライアント数が増加してサーバへのコネクションが多数発生すると，サーバやネットワークの負荷が増大する。また，各クライアントはサーバ上の同一データを共有することができず，それぞれのローカルなメモリにコピーしなければならないためオーバヘッドがある。ビジネスロジックがクライアント側，サーバ側に分散して配置されるため，仕様変更が頻繁に発生するクライアント側アプリケーションを容易に修正できない。

これらの点を解決するために最近注目されているアーキテクチャが3層クライアント・サーバ・モデルである(**図5**)[1]。3層アーキテクチャでは，クライアント層とサーバ層の間にビジネスオブジェクトを保持するドメイン層を導入する。この層をアプリケーションサーバ上に配置することによって，ビジネスオブジェクトを共有し，ビジネスロジックを高速に処理することができる。こうしてクライアントとデータベース・サーバの直接のコネクションを分離し，データ処理の一元化とビジネスロジックの集中を図ることができ，システムの応答性やアプリケーションの保守性が大きく向上する。

3.3 ネットワーク

ロンドン，ニューヨークなど主要海外拠点との連携による取引きのグローバル化に伴って，高速かつ安定な通信サービスが必要となっており，ATM(A synchronous Transfer Mode)などの高速通

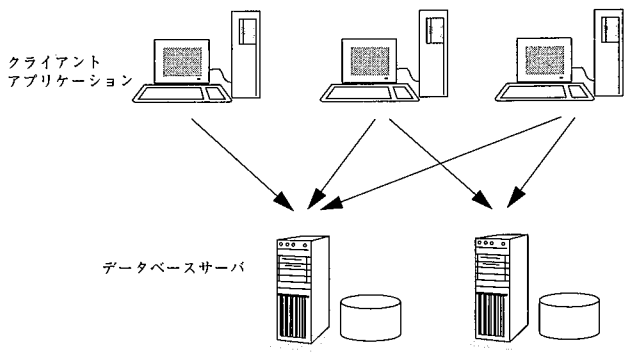

図4 クライアント・サーバ・アーキテクチャ

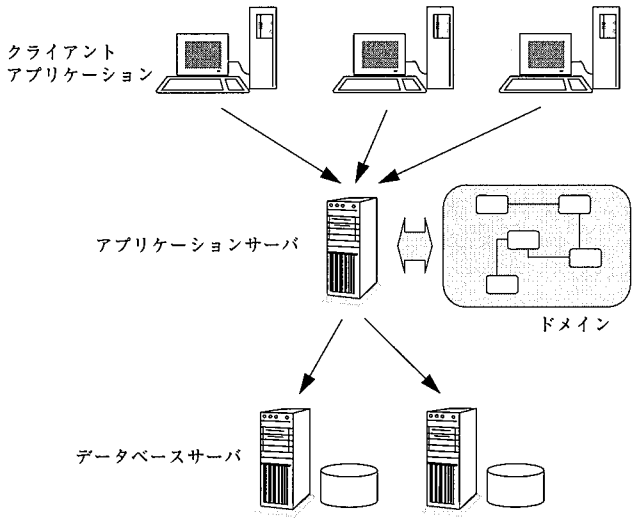

図5　3層構造のクライアント・サーバ・アーキテクチャ

信技術や分散オブジェクト技術に基づく通信ミドルウェアが活用されている。たとえば，米国New Era of Networks, Inc.や(株)インテリジェント・ウェイブより金融アプリケーション向けの通信ミドルウェアが提供されている。また，電子商取引を始めとして，WWW(World Wide Wed)ベースのインターネット，イントラネット技術も実用に供されるようになってきた。

3.4　データベース

金融情報系システムにおいては関係データベースシステム(RDBMS)が主流である。金利や債券などの取引入力にあたってはRDBMSによるトランザクション処理で十分な応答速度が得られる。しかし為替取引のように瞬時の応答が必要な場合や，日中のリスク管理のためにフロントからミドルにリアルタイムで取引データを配信しなければならない場合には，トランザクションモニタを使用して，リアルタイムの応答性を確保する必要がある。また，数百万件にのぼる取引データを対象とする夜間バッチ処理では，取引データの必要カラムのみを抽出したテーブルを別途作成するなどの工夫が必要である。

金融分野においては，24時間稼動などシステムの信頼性，データの可搬性などの点から，オブジェクト指向データベース(ODBMS)の普及度は高くない。しかし一方では，オブジェクト指向言語との親和性の良さや，RDBMSに比べ高速な応答性が得られることから，フロント取引入力データの一時入力バッファとしてObject Stoneを適用した事例もあるなど，用途に応じた活用が期待されている。

3.5　アプリケーション・フレームワークとデザインパターン

短期間でシステムを開発する最大のポイントは，コードやライブラリの再利用である。分析設計段階において再利用性の検討を織り込むとの対価を払い，かつオブジェクト指向技術を適用して再利用，拡張が可能な金融商品クラスライブラリを作成すれば，システムの生産性に大きく寄与することは先に述べた通りである。このようにして作成されたクラスライブラリなど各種コンポーネントを結合させ，アプリケーションの構築を容易にするための枠組みとなるものがアプリケーション・フレームワークである[2]。

図6はアプリケーション・フレームワークを利用して市場リスク管理システムを構築した場合のイメージ図である。まず，業務ルール，金融商品モデル，市場リスク管理モデルなど，基盤となる各種コンポーネントを組み合わせて，市場リスク管理システムのフレームワークを構築する。そしてこの市場リスク管理システム・フレームワーク上に，金融機関個別の業務形態に応じた独自の機能を付加することにより，その金融機関に適合した市場リスク管理アプリケーションを構築することが可能となる。他金融機関向けのシステムを開発する場合には，このフレームワークを再利用することによって，スクラッチ開発においても開発工期の大幅な削減が期待できる。たとえば米国Forte Software Inc.のフレームワークは，金融に限らず多くの分野で適用された実績がある。

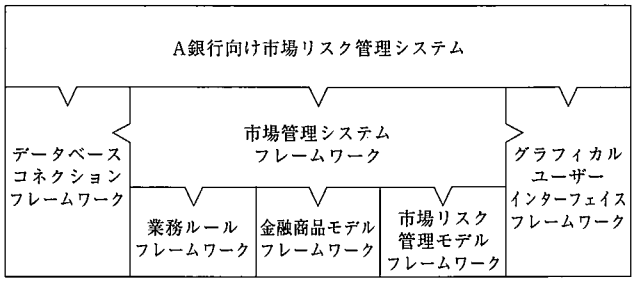

図6　アプリケーション・フレームワークを利用したシステム開発

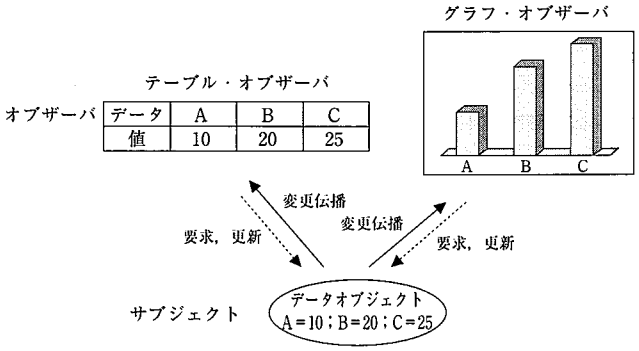

図7　オブザーバ・デザインパターン

このようなアプリケーション・フレームワークやコンポーネントを実装するにあたって，オブジェクト指向技術者の設計・実装ノウハウを広く収集し,カタログ化したものがデザインパターンである[3]。**図7**のオブザーバは，オブジェクト間の依存関係を実装するためのパターンである。テーブル及びグラフのオブザーバはサブジェクトに対し自分を登録しておくことで，サブジェクトのデータが更新された場合,自動的に更新伝播がそれぞれのオブザーバに通知される。また最近では分析モデルのパターン抽出とカタログ化も試みられており,金融業務アプリケーションの分析パターンも報告されている[1]。

4. 実施例

4.1　都市銀行向けオフバランスリスク管理システム

本節では，前節に述べたITの集大成ともいえる金融情報系システムの事例として，都市銀行向けオフバランスリスク管理システムを紹介する[4]。システム構成概念を**図8**に示す。本システム開発では，通信基盤ミドルウェアやアプリケーション・フレームワークをスクラッチから開発し，またアプリケーションの開発プロセスではプロトタイピング手法を適用した。システムは2層クライアント・サーバ・モデルをベースとしており，金融ドメインモデルはサブシステム化され，かつ拡張性，実行効率の両面を重視した実践的なモデルとなっている。本システムはアプリケーションだけでも250万行にものぼるが，かかる大規模なシステムに対して全面的にオブジェクト指向技術を適用した世界初の事例の一つといえる。

本システムのアプリケーション・フレームワークの構造を**図9**に

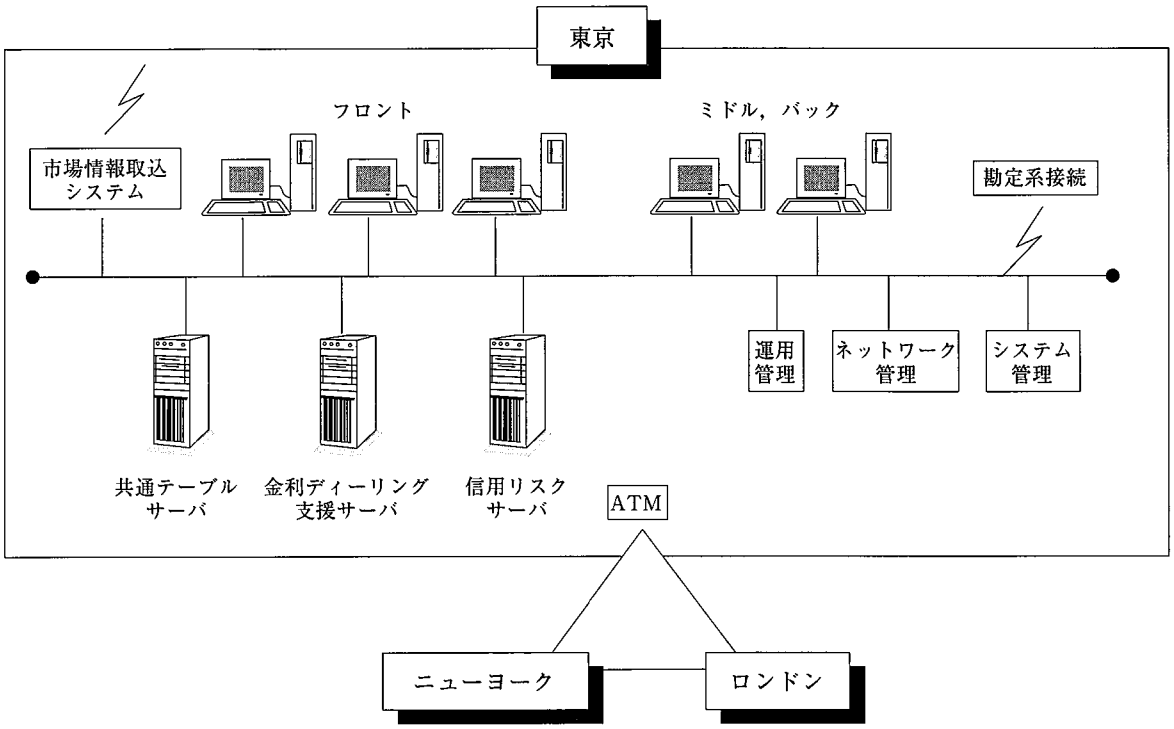

図8　都市銀行向けオフバランスリスク管理システムのイメージ図

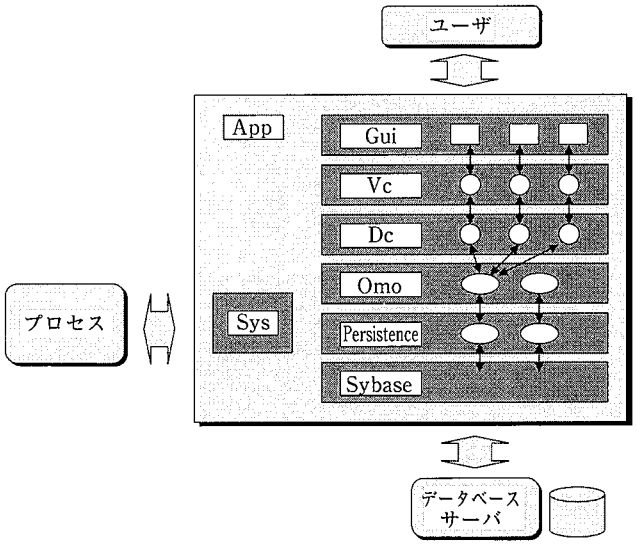

図9　アプリケーション・フレームワークの構築例

示す。各層の主な機能は，(1)App(アプリケーション全体のライフサイクル制御，Sys経由によるプロセス間通信)，(2)Gui(GUI部品のクラス化)，(3)Vc(画面コールバックの制御)，(4)Dc(データ変換とアプリケーションロジックの提供)，(5)Omo(ドメインオブジェクトの管理)，(6)Persistence(RDBMSとドメイン間のインターフェイス)，及び(7)Sybaseである。

Gui，Vc，Dc，Omo 層の各部品は，オブザーバ・デザインパターンを使用して設計されており，画面のフィールドデータやドメインオブジェクトの内容が変更された場合，自動的に更新情報が相互に伝播する仕組みになっている。またOmo層は3層構造アーキテクチャのドメイン層に対応するもので，ビジネスロジックを集中的に実装している。ただしOmo層はクライアント・アプリケーションの一部として結合され，システム全体としては2層構造である。また，Omo層は Persistence が発行するSQLを最小限に抑制し，システム全体としてのパフォーマンスを向上させる機能を併せもつ。

アプリケーション・フレームワークをベースとするシステム開発では，実装者の導入教育に時間を要するものの，アプリケーションの設計・実装に一貫した制約を持たせることで，実装者間のコードのばらつきを排除でき，品質，保守性が大きく向上している。

4.2　証券向け金融業務システム構築用アプリケーション・フレームワーク

第二の事例として，米国Iris Financial Engineering & Systems,Inc.が開発し販売している金融業務システム開発用ミドルウェアのパッケージを紹介する。本ミドルウェアはオブジェクト指向技術をベースとする統合ツールであり，クライアント・サーバ・アーキテクチャの金融業務アプリケーションを容易に構築することができる。本ミドルウエアが提供する機能は，オブジェクトモデリング機能，分散環境でのリアルタイムデータ配信機構，ビジネス及びGUIオブジェクト用の基本クラスライブラリ，GUI構築コンポーネント，アプリケーションビルダ，そして，金融商品を分析するためのクラスライブラリなどである。業務モデルは各金融機関の業務形態にあわせて，モデリング機構を使用して個別に作成する。また，金融クラスライブラリも独自拡張が可能である。

図10は本ミドルウエアを使用した3層クライアント・サーバ・アプリケーションの概念図である。金融のビジネスオブジェクトはアプリケーションサーバ上に保有され，各クライアントが必要とするオブジェクトはキャッシュ機構により各プロセス間で共有される。データベースへのコミットに伴う他のプロセスへの変更伝播は，本ミドルウエアの標準コンポーネント(データベース・ゲートウェイ，キャッシュ管理，通信管理)を通じて自動的に行われる。GUIのみをもつクライアントはアプリケーションビルダにより容易に構築できるため，システム設計者はアプリケーションサーバで実行されるアプリケーション及びビジネスロジックの開発に集中すればよい。

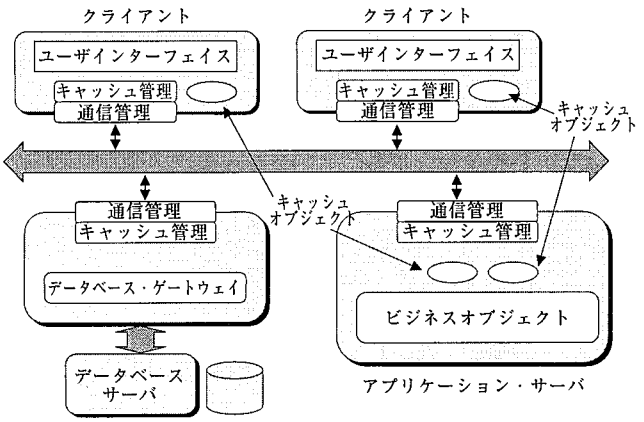

図10 IRISによる3層構造アプリケーションの実現例

5. 結　言

金融情報系システムにおけるITについて，クライアント・サーバ・モデルとオブジェクト指向技術を主体に概観した。今後の動向としては，C++言語の標準テンプレートライブラリ(STL)や分散オブジェクト(CORBA)などの標準に完全に準拠し，かつ実用に耐えうる本格的なミドルウェアの出現が期待され，ベンダーに独立な標準化技術を使用したシステム構築が可能となりつつある。また，アプリケーションの保守性向上のため，EUC(End User Computing)を中心にWWWベースのシステムが増えてくるであろう。その場合，習得が容易なJava言語が普及すると思われる。すでに米国ではJavaベースの金融業務アプリケーションが出現している。このように，金融情報系システムの開発においては，標準化されたオブジェクト指向技術をベースとするクライアント・サーバ・システムが今後の主流となるであろう。

参照文献

1) Fowler,M.：Analysis Patterns: Reusable Object Models. 1st ed. Reading, MA, Addison Wesley, 1997
2) Rogers,G.F.：Framework-Based Software Development in C++. 1st ed. Upper Saddle River, NJ, Prentice-Hall, 1997
3) Gamma,E. et. al.：Design Patterns: Elements of Reusable Object-Oriented Software. 1st ed. Reading, MA, Addison Wesley, 1995
4) 南　悦郎　ほか：Object World Expo/Tokyo 1996, 東京, 1996-10