# Client Server Architecture

## Course Material

## CONTENTS

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Topic</th>
<th>Contents</th>
</tr>
</thead>
</table>
| 1.    | Introduction to Client/Server | 1.1 Introduction to Client/Server  
1.2 Characteristics of the Client And the Server  
1.3 Merits and Demerits of the Client Server |
| 2.    | Client/Server Architecture And Servers | 2.1 Types of Servers  
2.2 ORB  
2.3 Client Server Architectures  
   2.3.1 The knowledge about the Fat Servers And the Fat Clients  
   2.3.2 The Architectures  
2.4 Stored Procedure  
2.5 Remote Procedure Call (RPC) |
| 3.    | Client Side Services | 3.0 Services  
   3.0.1 Introduction  
   3.0.2 Role of the client and its services  
   3.0.3 Request for Services  
   3.0.4 Redirection  
3.1 Print Services  
3.2 Remote Services  
3.3 Utility Services  
3.4 Message Services  
3.6 Network Services  
3.6 Application Services  
3.7 Database Services  
3.8 Dynamic Data Exchange (DDE)  
3.9 Object Linking and Embedding (OLE)  
3.10 Client Tools  
3.11 GUI Clients  
3.12 Non-GUI Clients  
3.13 OOUI (Object Oriented User Interface) Clients |
| 4. | **Server Side Services** | 4.1 Server Functionality  
4.2 Request Processing  
4.3 Print Services  
4.4 Database Services  
4.6 Security Services  
4.6 File Services  
4.7 Communication Services |
|---|---|---|
| 5. | **Network and Protocol Basics** | 5.0 Communication Network  
5.1 Local Area Network  
5.2 Metropolitan Area Network  
5.3 Wide Area Network  
5.4 Network Structure  
5.6 OSI Model  
5.6 TCP/IP Architecture  
5.7 TCP/IP Protocols |
| 6. | **Hardware** | 6.1 Introduction  
6.1.1 The Hardware Issues  
6.1.2 Connectivity  
6.1.3 Communication Server  
6.2 Hardware/Network Acquisition  
6.3 Different types of Computing Devices  
6.3.1 PC-Level Processing Units  
6.3.2 Macintosh  
6.3.3 Notebooks  
6.3.4 Pen  
6.3.5 UNIX Workstation  
6.4 Server Hardware  
6.5 Data Storage  
6.5.1 Magnetic Disk  
6.5.2 Mirrored Disk  
6.5.3 RAID-Disk Array  
6.5.4 Tape  
6.5.5 Optical Disks  
6.5.6 CD-ROM  
6.5.7 WORM  
6.5.8 Erasable Optical  
6.5.9 Erasable Optical  
6.5.10 Network Interface Cards (NICs)  
6.5.11 Power Protection Devices  
6.5.12 Uninterruptible Power Supply (UPS) |
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Content</th>
</tr>
</thead>
</table>
| 7       | Application development management issues | 7.1 Need to Improve Technology Professional Productivity  
7.2 Platform Migration  
7.3 Client/Server Systems Development Methodology  
7.4 Project Management CASE |
| 8       | CORBA | 8.1 Common Request Broker Architecture (CORBA)  
8.1.1 OMG (Object Management Group)  
8.1.2 CORBA .......... what is it?  
8.1.3 who is using CORBA already?  
8.1.4 what is CORBA good for?  
8.1.5 ORB and CORBA Architecture  
8.1.6 Technical Details of CORBA  
8.2 CORBA Object Services  
8.3 CORBA Common Facilities  
8.4 CORBA 3.0: Next Generation  
8.6 CORBA Style Interfaces  
8.6 DCOM Objects  
8.6.1 Component Object Model (COM) History  
8.6.2 About COM  
8.6.3 What is DCOM?  
8.6.4 Why use DCOM?  
8.6.5 Benefits of DCOM  
8.6.6 What is not good in DCOM?  
8.6.7 Basic Concepts of DCOM  
8.7 COM Servers |
| 9       | Appendix | Network Operating System |
UNIT I
Introduction to Client/Server

1.1 Introduction to Client/Server
1.2 Characteristics of the Client and the Server
1.3 Merits and Demerits of the Client Server

1.1 Introduction
1.1.1 The Vision of the Client/Server Computing

We are in the midst of a fundamental change in both technology and its application. Organizations today expect to get more value from their investments in technology.

Opportunities are available to organizations and people who are ready and able to compete in the global market. A competitive global economy will ensure obsolescence and obscurity to those who cannot or are unwilling to compete. All organizations must look for ways to demonstrate value. We are finally seeing a willingness to rethink existing organizational structures and business practices.

Organizations are aggressively downsizing even as they try to aggressively expand their revenue base. There is more willingness to continue
improvement practices and programs to eliminate redundancy and increase effectiveness. Organizations are becoming market-driven while remaining true to their business vision.

_client/server computing is the most effective source for the tools that empower employees with authority and responsibility._

Workstation power, workgroup empowerment, preservation of existing investments, remote network management, and market-driven business are the forces creating the need for client/server computing.

_the Client/server computing is an irresistible movement that is reshaping the way computers is being used._ Though this computing is very young it is already in full force and is not leaving any area and corner of the computer industry untouched. The Client/Server application development requires hybrid skills that include transaction processing, database design, communication experience, graphical user interface design, and being Internet Savvy. The more advanced applications require knowledge of distributed objects and component infrastructures.

Most client/server solutions today are PC LAN implementation that is personalized for the group that uses them.

The _Client/Server Computing_ has changed the way in which computers are being used. This has a unique feature of having a strong foothold on the entire spectrum of the computer industry.

For the “_PC my World_” lot, the Client/Server Computing means - scrapping of every mainframe that cannot fit on the desktop and thus end of the host-centric computing.

For the “_Diehard fans of the Mainframes_”, the Client/Server Computing means unleashing the rebirth of the networked mainframes that will bring every PC in the enterprise block to the fold.

For “_Men in between_”, it is a new era of Internet based co-existence and openness in which all can play.

Let us now know what a client is and what is a server?
• **Client** - A client is a single-user workstation that provides presentation services and the appropriate computing, connectivity and the database services and the interfaces relevant to the business need.

• **Server** - A server is one or more multi-user processors with shared memory providing computing, connectivity and the database services and the interfaces relevant to the business need.

The Client/Server computing is an environment that satisfies the business need by appropriately allocating the application processing between the client and the server processors.

The protocol is the client requests the services from the server; the server processes the request and returns the result to the client. The communication mechanism is a message passing InterProcess communication (IPC) that enables the distributed placement of the client and server processes.

The Client/Server is the generic model and fits what is known in the industry as the “cooperating processing” or “peer-to-peer”. The client/server computing is fundamentally platform independent.

The user of an application wants the functionality (business) it provides; the computing platform provides access to this business functionality. There are no benefits but a considerable amount of risk of exposing the platform to the users.

The changes in the platform and the underlying technology should be transparent to the user, as it is understood that the systems built with transparency to the technology, for the entire user offer the highest probability of solid ongoing return for the technology investment.

It is also easily demonstrable that is the developers become aware of the target platform, development will be bound to that platform and they will use special tricks and feature found in that specific platform. The figure 1.1 shown below depicts the modern client/server architecture.
1.2 Characteristics of the Client and the Server

The clients and the servers are the logical entities that work together over a network to accomplish a task.

The distinguishing characteristics of the Client/Server systems are:

1. **Service**: The client/server is primarily a relationship between processes running on separate machines. The server process is a provider of services. The client is a consumer of services. In essence, client/server provides a clean separation of function based on the idea of service.

2. **Shared Resources**: A server can service many clients at the same time and regulate their access to shared resources.
3. **Asymmetrical protocols:** There is a many-to-one relationship between the clients and the server. Clients always initiate the dialog by requesting a service.

   Servers are passively awaiting request from the clients. In some cases a client may pass a reference to a callback object when it invokes a service. This lets the server call back the client. So the client becomes a server.

4. **Transparency of location:** The server is a process that can reside on the same machine as the client or on a different machine across a network. Client/Server software usually masks the location of the server from the clients by the redirecting the service calls when needed. A program can be a client, a server, or both.

5. **Mix-and-match:** The ideal client/server software is independent of hardware or operating system software platforms. You should be able to mix-and-match client and server platforms.

6. **Message-based exchanges:** Clients and servers are loosely coupled systems that interact through a message-passing mechanism. The message is the delivery mechanism for the service request and replies.

7. **Encapsulation of services:** The server is a specialist. A message tells a server is requested; it is then up to the server to determine how to get the job done. Servers can be upgraded without affecting the clients as long as the published message interface is not changed.

8. **Scalability:** Client/Server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means either migrating to a larger and faster server machine or distributing the processing load across multiple servers.

9. **Integrity:** The server code and server data is centrally managed, which results in cheaper maintenance and the guarding of shared data integrity. At the same time, the clients remain personal and independent.
1.3 Merits and Demerits of the Client Server

The Merits of Client/Server Computing

Client/server computing provides the capability to use the most cost-effective user interface, data storage, connectivity, and application services. Frequently, client/server products are deployed within the present organization but are not used effectively.

The client/server model provides the technological means to use previous investments in concert with current technology options. Organizations see opportunities to use technology to provide business solutions. Service and quality competition in the marketplace further increases the need to take advantage of the benefits available from applications built on the client/server model.

Client/server computing in its best implementations moves the data-capture and information-processing functions directly to the knowledgeable worker—that is, the worker with the ability to respond to errors in the data, and the worker with the ability to use the information made available.

1. Enhanced Data Sharing

Data that is collected as part of the normal business process and maintained on a server is immediately available to all authorized users.

The use of Structured Query Language (SQL) to define and manipulate the data provides support for open access from all client processors and software.

SQL grants all authorized users access to the information through a view that is consistent with their business need. Transparent network services ensure that the same data is available with the same currency to all designated users.
2. *Integrated Services*

In the client/server model, all information that the client is entitled to use is available at the desktop. There is no need to change into terminal mode or log into another processor to access information.

All authorized information and processes are directly available from the desktop interface. The desktop tools—e-mail, spreadsheet, presentation graphics, and word processing—are available and can be used to deal with information provided by application and database server’s resident on the network.

Desktop users can use their desktop tools in conjunction with information made available from the corporate systems to produce new and useful information.

3. *Sharing Resources among Diverse Platforms*

The client/server computing model provides opportunities to achieve true open system computing. Applications may be created and implemented without regard to the hardware platforms or the technical characteristics of the software.

Thus, users may obtain client services and transparent access to the services provided by database, communications, and applications servers.

Operating systems software and platform hardware are independent of the application and masked by the development tools used to build the application.

In this approach, business applications are developed to deal with business processes invoked by the existence of a user-created "event."

4. *Data Interchangeability and Interoperability*

SQL is an industry-standard data definition and access language. This standard definition has enabled many vendors to develop production-class database engines to manage data as SQL tables. Almost all the development
tools used for client/server development expect to reference a back-end database server accessed through SQL.

Network services provide transparent connectivity between the client and local or remote servers. With some database products, such as Ingres Star, a user or application can define a consolidated view of data that is actually distributed between heterogeneous, multiple platforms.

Systems developers are finally reaching the point at which this heterogeneity will be a feature of all production-class database engine products. Most systems that have been implemented to date use a single target platform for data maintenance. The ability to do high-volume updates at multiple locations and maintain database integrity across all types of errors is just becoming available with production-level quality performance and recovery.

Systems developed today that use SQL are inherently transparent to data storage location and the technology of the data storage platform. The SQL syntax does not specify a location or platform.

This transparency enables tables to be moved to other platforms and locations without affecting the application code. This feature is especially valuable when adopting proven, new technology or if it makes business sense to move data closer to its owner.

5. **Location Independence of Data and Processing**

We are moving from the machine-centered computing era of the 1970s and 1980s to a new era in which PC-familiar users demand systems that are user-centered. Previously, a user logged into a mainframe, mini-, or micro application. The syntax of access was unique in each platform. Function keys, error messages, navigation methods, security, performance, and editing were all very visible. Today's users expect a standard "look and feel." Users log into an application from the desktop with no concern for the location or technology of the processors involved.

6. In most cases, client-server architecture enables the roles and responsibilities of a computing system to be distributed among several independent computers that are known to each other only through a network.
This creates an additional advantage to this architecture: greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change. This independence from change is also referred to as *encapsulation*.

7. All the data is stored on the servers, which generally have far greater security controls than most clients. Servers can better control access and resources, to guarantee that only those clients with the appropriate permissions may access and change data. Since data storage is centralized, updates to that data are far easier to administer than what would be possible.

8. Many mature client-server technologies are already available which were designed to ensure security, 'friendliness' of the user interface, and ease of use. It functions with multiple different clients of different capabilities.

**The Demerits of Client/Server Computing**

*(Comparisons with the P2P)*

1. Traffic congestion on the network has been an issue. As the number of simultaneous client requests to a given server increases, the server can become severely overloaded. Contrast that to a P2P network, where its bandwidth actually increases as more nodes are added, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network.

2. The client-server paradigm lacks the robustness of a good P2P network. Under client-server, should a critical server fail, clients’ requests cannot be fulfilled. In P2P networks, resources are usually distributed among many nodes. Even if one or more nodes depart and abandon a downloading file, for
example, the remaining nodes should still have the data needed to complete the download.

UNIT 2

Client/Server Architecture and Servers

2.1 Types of Servers
2.2 ORB
2.3 Client Server Architectures
   2.3.1 about the Fat Servers and the Fat Clients
   2.3.2 The Architectures
2.4 Stored Procedure
2.5 Remote Procedure Call (RPC)

2.1 Types of Server

i. File Server

   • File Servers are useful for sharing information across the network
   • The client passes a request for file records over a network to the file server.
• This is the most primitive type of data service used for exchanging messages over the network to find the requested data.
• The file servers provide access to the remote server processors. In the typical implementation the software, shared data, databases and backups are stored on disk, tape and optical storage devices that are managed by the file server.

**ii. Database Server**

• The client passes the SQL requests as messages to the database server; the result of each SQL command is returned over the network.
• The code, which processes the SQL request and the data, reside in the same machine, the server uses its own processing power to find the requested data back to the client, instead of passing all the records back to the client. This results in a much more efficient use of the distributed processing power.
• Note that the application code resides on the client; thus you must either write code for the client or you can buy a shrink-wrap query tool.
• The database servers provide the foundation for decision-support systems and also provide key role in data warehousing.
iii. Transaction Servers

The client can invoke remote procedure or services that reside on the server with an SQL database engine using the transaction server.

- The network exchange consists of a single request/reply. The SQL statements either all succeeded or fail as a unit.
- These grouped SQL statements are called transactions.
- With a transaction server you create the client/server application by writing the code for both the client and the server components.
- The client component usually includes a Graphical User Interface (GUI). The server component consists of SQL transaction against a database. These applications are called Online Transaction Processing or OLTP.
- The OLTP are mission-critical applications that require a very less response time (1-3 sec).
- The OLTP applications also require tight controls over the security and integrity of the database.
- It has two forms
  - TP-Lite: based on the stored procedures provided by database vendors.
TP-Heavy: bases on the TP Monitors which is provided by OLTP vendors.

**iv. Groupware Servers**

- It involves the management of semi-structured information such as text, image, mail, bulletin boards and the flow of work.
- These client/server system places people in direct contact with other people. *Best examples are Lotus Notes and Microsoft Exchange.*
- Specialized groupware software can be built on top of a vendor’s canned set of client/server API’s. In many cases, applications are created using a scripting language and form-based interfaces provided by the vendor. Now many groupware products use e-mail as their standard messaging middleware. Also, Internet is quickly becoming the middleware platform of choice for groupware.
2.1 d Client /Server with Groupware Servers

v. Object Application Servers.

- The client/server application is written as a set of communicating objects with an object server.

- The client object uses the Object Request Broker (ORB) to communicate with the server objects.

- The ORB locates an instance of that object server class, invokes the requested method and returns the results to the client object.

- The server objects must support the concurrency and sharing aspects. The ORB and a new generation of CORBA application servers bring it all together.

- The commercial ORB’s that comply with the Object Management Group’s CORBA standard include Iona’s Orbix, Inprise’s VisiBroker, ICL’s DAIS, JavaSoft’s Java IDL, BEA’s ObjectBroker, IBM’s SOM and Expersoft’s PowerBroker.

- CORBA is also the basic foundation technology for the Enterprise JavaBeans component model. The new generation CORBA application servers are also called as the Object Transaction Monitors (OMTs) provide server-side component coordination services.

- Microsoft hat Distributed Component Object Model (DCOM) as an competitor for CORBA; Where DCOM is the foundation technology for Microsoft’s enterprise software and also it s ActiveX component
model. The Microsoft Transaction Server (MTS) is the application server for ActiveX components; it is also called as COM+.

- **Web Application Server**
  Web application servers are a new class of Internet software. They combine the standard HTTP servers with server-side component frameworks. Functionally they are very similar to the object servers.

- **This model of the client/server consists of thin, portable, and “universal” client that talk to the super fat servers.** Here the Web server returns documents when clients ask for them by name. The clients and the servers communicate using an RPC-like protocol called HTTP, which defines a simple set of commands, parameters which are passed as strings, with no provision for typed data.

- In case of Microsoft, the MTS distributed object server is also the Web Application Server. Whereas in case of CORBA/Java; Enterprise JavaBeans has become common currency of Web Application servers.

- Some of these servers also provide the COM/CORBA bridges.

- The examples of the CORBA/Java Web application servers are Netscape /Kiva’s Application Server, BEA’s WebLogic, Bluestone’s Sapphire Web, IBM’s WebSphere, SilverStream’s SilverStream Application Server 2.0, Novera’s jBusiness Application Server etc.
2.1e Client/Server with Distributed Objects

2.2 ORB

An object request broker (ORB) is a middleware technology that manages communication and data exchange between objects.

ORB is the object bus. It lets the objects transparently make request to - and receive responses from – other objects located locally or remotely.

The client is not aware of the mechanisms used to communicate with, activate or store the server objects. A CORBA ORB provides a very rich set of distributed middleware services.

An ORB is much more sophisticated than alternative forms of client/server middleware including the traditional Remote Procedure Calls (RPC’s), Message-Oriented Middleware (MOM), database stored procedures, and peer-to-peer services. CORBA is the best client/server middleware pipe ever defined.

ORBs promote interoperability of distributed object systems because they enable users to build systems by piecing together objects- from different vendors- that communicate with each other via the ORB.
ORB technology promotes the goal of object communication across machine, software, and vendor boundaries. The functions of an ORB technology are

- Interface definition
- Location and possible activation of remote objects
- Communication between clients and object

An object request broker acts as a kind of telephone exchange.

It provides a directory of services and helps establish connections between clients and these services is illustrated in the figure 2.2

Fig 2.2 Object Request Broker

The ORB must support many functions in order to operate consistently and effectively, but many of these functions are hidden from the user of the ORB.

**Responsibilities of the ORB**

1. *Providing Illusion of Locality:*

   It is the responsibility of the ORB to provide the illusion of locality, in other words, to make it appear as if the object is local to the client, while in reality it may reside in a different process or machine.
Thus the ORB provides a framework for cross-system communication between objects. This is the first technical step toward interoperability of object systems.

2. *Hide the Implementation Details:*

The next technical step toward object system interoperability is the communication of objects across platforms. An ORB allows objects to hide their implementation details from clients. This can include programming language, operating system, host hardware, and object location.

Each of these can be thought of as a "transparency," and different ORB technologies may choose to support different transparencies, thus extending the benefits of object orientation across platforms and communication channels.

There are many ways of implementing the basic ORB concept; for example, ORB functions can be compiled into clients, can be separate processes, or can be part of an operating system kernel. These basic design decisions might be fixed in a single product; or there might be a range of choices left to the ORB implementer.

There are two major ORB technologies:

- The Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) specification
- Microsoft's Component Object Model (COM).

### 2.3 Client Server Architecture

#### 2.3.1 The Knowledge about the Fat Servers and the Fat Clients

We know that the client/server models can be distinguished by the service they provide and how the distributed application is split between the client and the server.

In these lines we have

1. Fat Server Model
2. Fat Client Model

#### 1. The Fat Server Model
Client/Server Architecture

- Place more functions on the server
- Usually used for the mission-critical applications.
- Applications are easier to manage on the network as most of the work is done on the server.
- The fat servers create abstract level of services by which the network interchange is minimized.
- The Transaction Servers and the Object Server embed the concept of encapsulation of database by exporting the procedure/ methods, which act on the data instead of the raw data.
- The client interacts with such fat servers using the remote procedure call.
- The examples are the Groupware, Transaction Servers, and Web Servers.

2. Fat Client Model

- Places more function on the client. In a client/server architecture, a client that performs the bulk of the data processing operations. The data itself is stored on the server.
- They are the traditional form of the client/server systems
- They are generally used for decision support and personal software
- They lead to the creation of the front-end tools and applications.
- The best places are – the file server and the database server models where the client knows how the data is organized and stored on the server.

We must know that in the real/actual working environment both fat server and the fat client may coexist in one application.

2.3.2 The Architectures

A network architecture in which each computer or process on the network is either a client or a server.

Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.
Another type of network architecture is known as a *peer-to-peer* architecture because each *node* has equivalent responsibilities. Both client/server and *peer-to-peer architectures* are widely used, and each has unique advantages and disadvantages. Client-server architectures are sometimes called *two-tier architectures*.

The gurus of the client/server architecture do not use the terms fat client and the fat servers on the contrary they refer to them are 2-tier, 3-tier and N-tier client/server architecture. This is the means by which they are functionally split. The functional units comprise of user interface, business logic and the shared data.

1. **2-tier Client/Server Architecture**

Two tier software architectures were developed in the 1980s from the file server software architecture design. The two-tier architecture is intended to improve *usability* by supporting a forms-based, user-friendly interface.

The two-tier architecture improves *scalability* by accommodating up to 100 users (file server architectures only accommodate a dozen users). It also proves *flexibility* by allowing data to be shared, usually within a homogeneous environment.

The two-tier architecture requires minimal operator intervention, and is frequently used in non-complex, non-time critical information processing systems.
Two tier architectures consist of three components distributed in two layers: client (requester of services) and server (provider of services).

The three components are

1. User System Interface (such as session, text input, dialog, and display management services)
2. Processing Management (such as process development, process enactment, process monitoring, and process resource services)
3. Database Management (such as data and file services)

The two-tier design allocates the user system interface exclusively to the client. It places database management on the server and splits the processing management between client and server, creating two layers.

The application logic may be present at the client side within a user interface or it may be present within the database on the server or on the both. It is most popular because of its simplicity.

These applications can be quickly built by using and visual builder tools; which can be used for developing applications for decision support system of small-scale groupware or you may build a simple web publishing applications.
But the real problem arises only when you deploy them beyond the departmental LAN. Typically the applications that worked perfectly well in prototypes and small installations failed for large-scale productions. It actually went through a transition phase, where it grew beyond the departmental LAN’s. Thus the complex world is now faced by the 3-tier and the N-tier applications.

2. **3-tier Client/Server Architecture**

The three-tier software architecture emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two-tier architecture) by providing functions such as queuing, application execution, and database staging.

The three-tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two-tier architecture) increased **performance**, **flexibility**, **maintainability**, **reusability**, and **scalability**, while hiding the complexity of distributed processing from the user.

They are also easy to manage and deploy the network and most of the code runs on the server. The protocol of interaction between the client and the server is as follows:

*The client calls for the business logic on the server, the business logic on the behalf of the client accesses the database.*

The 3-tier substitutes a few server calls for many SQL queries and updates so it performs much better than 2-tier. A three-tier distributed client/server architecture includes a user system where user services (such as session, text input, dialog, and display management) reside.

The middle tier provides process management services (such as process development, process enactment, process monitoring, and process resourcing) that are shared by multiple applications.
The third tier provides database management functionality and is dedicated to data and file services that can be optimized without using any proprietary database management system languages.

**3-tier to N-tier**

In the 3-tier applications, the middle tier is generally not a monolithic program but is implemented as a collection of components that are initiated by several client-initiated business transaction. One component can call other components to help it implement a request; also some components may act as gateways which may encapsulate legacy applications running on mainframe.
The component-based application provides the following advantages over the monolithic applications.

1. **You can develop big applications in small steps**
   Large mission-critical application projects may be divided into small component projects. The component enriches them for a small-team and incremental development approach.

2. **You can reuse components**

3. **Clients can access data and functions easily and safely.**
   The client sends request to the components to execute on their behalf. The clients do not need to know which database is being accessed to execute the request; also they do not need to know if the request was sent to another component or application for execution. The server component encapsulates the details of the application logic and thus raises the level of abstraction.

4. **Custom applications can incorporate off-the-shelf components**
   Today’s trend is to buy ready-made components that are packaged as applications. They can also be mix and matched with different software vendors; but a word of caution is the “Mix-and-Match “capability needs some semantic glue.

**When can /should you use 3-tier?**

According to *Gartner Group*’s for the smaller projects – 2-tier applications are easier to develop than 3-tier, however as the application becomes more complex, 2-tier applications become exponentially harder to develop.

According to Gartner Group, *if your application has any of the following characteristics then you may choose 3-tier architecture:*

1. Many applications services/classes (around or more than 60)
2. Application programmed in different languages or written by different organizations
3. An application life longer than 3 years.
4. If there are more than 60,000 transactions per day or more than 300 concurrent users on the same system accessing the same database.
5. Two or more heterogeneous data sources like two DBMS or a DBMS and a file system.
Intergalactic Client/Server

The client/server applications stand at a new threshold brought on by the exponential increase of low-cost bandwidth on Wide Area Network for example, the Internet and AOL.
1. A new generation of Web-enabled desktops and devices. This new threshold marks the beginning of a transition from 2-tier Ethernet client/server to N-tier intergalactic client/server.
2. The client and the server do not have to be co-located on the same campus; but they can be a world away. This is called as the “Irrelevance of proximity”.

Client/Server Building Blocks

When we think of client/server systems there are few questions, which come into our mind, like
1. How is the application split between the client and the server?
2. What function goes into the client and what function goes in the server?
3. Can the client/server model accommodate business of all sizes?
4. Can the client/server play at home?
5. How are the new tribes of nomadic laptop users brought into the client/server fold?

The answer for all these…
The client/server model accommodates all types of users, it is simple and it works well with today’s technologies. Let us now see which are the 3 building blocks of the client/server system, which meet a wide spectrum of client/server needs from the tiny to the intergalactic.

The 3 building blocks are
1. Client
2. Server
3. Middleware
Now let us see are they used for the following four situations:

1. **Client/Server for tiny shops and nomadic tribes**
   - Here the client, middleware software and most of the business services run on the same machine.
   - Suggested for implementation is home office, one-person shops and mobile user with well-endowed laptops.

2. **Client/Server for small shops and departments**
   - This is the classic Ethernet client/server building block implementation.
   - Used in small shops, departments and branch offices.
   - This is one of the predominant forms of client/server today.

3. **Client/Server for intergalactic enterprise**
   - This forms the multiserver building-block implementation of client/server.
   - The server presents a single-system image to the client.
   - They can be spared throughout the enterprise, but they can be made to look like they are part of the local desktop.
   - This implementation meets the initial needs of intergalactic client/server computing.

4. **Client/Server for a post-scarcity world**
   Transforms every machine in the world into both a client and a server. Personal agents on every machine will handle all the negotiations with their peer agents anywhere in the universe.

**Building Blocks in Nutshell**

Let us now view each of the building blocks in detail.

1. **The client building block**
   - Runs the client side of the application
   - It runs on the Operating System that provides a Graphical User Interface (GUI) or an Object Oriented User Interface (OOUI)
   - The thin clients require a Web Browser to download JavaBeans and applets on demand.
• This blocks the operating system, passes the buck to the middleware and lets it handle the non-local services.

• The client also runs a component of the Distributed System Management element (DSM). Where the DSM could be a simple agent on a PC or a front-end of the DSM application on a managing station.

2. The server building block
• Runs the server part of the application.
• We have 6 contending server platforms for creating the next generation of client/servers.
  o TP Monitors
  o SQL Database Servers
  o Groupware Servers
  o Object Servers
  o Web
• The server is operating system dependent to interface with the middleware.
• The server also runs a DSM component, which is once again a simple agent on a managed PC to the entire back-end of the DSM application.

3. The middleware building block
• It runs on both the client and the server sides of an application.

• This is further divided into 3 categories
  o Transport stacks
  o Network Operating Systems (NOS)
  o Service-specific Middleware

• This also has a DSM software component.

• Where middleware forms the nervous system of the client/server infrastructure.
2.4 **Stored Procedure**

- Stored Procedure is a named collection of SQL statements and procedural logic that is compiled, verified and stored in the server database.

- Sybase pioneered this concept of stored procedure in 1986 to improve the performance of SQL on networks.

- The major database vendors are now offering an RPC-like mechanism for executing functions that are stored in their databases. This mechanism is sometimes referred to as “TP-Lite” or “stored procedure”.

- A stored procedure is typically treated like any other database object and registered in the SQL catalog. The access to the stored procedure is controlled through the server’s security mechanisms.

- They are
  1. Used to enforce business rules and data integrity
  2. Perform system maintenance and administration functions
  3. To extend the database server’s functions
  4. Used to create the server side of application logic.
  5. It is also well suited for creating performance-critical applications known as Online Transaction Processing OLTP

- Typically these applications can
  1. Receive a fixed set of inputs from remote clients
  2. Perform multiple precompiled SQL commands against a local database
  3. Commit the work
  4. Return a fixed set of results.
  5. Stored Procedures are database-centric, RPC-like SQL entity that is persistent, shared and has a name,
  6. It reduces the network traffic and improves the response time.
  7. Provide better site autonomy because the remote modification of tables can only occur through locally executing programs.
8. Any changes are made to the tables need not be recompiled in all applications in the remote machines.

9. They provide better distribution of intelligence than static or dynamic SQL.

10. It provides a service-oriented interface that is well suited for OLTP applications.

Major Drawbacks of Stored Procedure

- They provide less ad hoc flexibility than remote dynamic SQL.
- They may perform very poorly if their plans are not refreshed/rebound to take advantage of the optimize statistics.

- They are totally non-standard. Results in number of problems, no two-vendor implementations are alike.

- There is no standard way to pass or describe the parameters.

- The language for describing the stored procedure is not portable across vendor platform.

- There is no transactional synchronization—i.e., two-phase commit between stored procedures; where each stored procedure is a separate transaction.

2.6 Remote Procedure Call (RPC)

Over the years, good programmers have developed modular code using structured techniques and subroutine logic. Today, developers want subroutines to be stored as named objects "somewhere" and made available to everyone with the right to use them.
Remote procedure calls (RPCs) provide this capability. RPCs standardize the way programmers must write calls, so that remote procedures can recognize and respond correctly.

The RPC hides the intricacies of the network by using the ordinary procedure call mechanism familiar to every programmer.

A client process calls a function on a remote server and suspends itself until it gets back the results. Here the parameters are passed like any ordinary procedure. The server receives the request, unpacks the parameters, calls the procedure, and sends the reply backs to the client.

While RPC makes life easier for the programmer, they pose a challenge for the NOS designers who supply the development tools and run-time environments.

If an application issues a functional request and this request is embedded in an RPC, the requested function can be located anywhere in the enterprise that the caller is authorized to access. The RPC facility provides for the invocation and execution of requests from processors running different operating systems and using hardware platforms different from that of the caller. Many RPCs also provide data translation services. The call causes dynamic translation of data between processors with different physical data storage formats. These standards are evolving and being adopted by the industry.
UNIT 3
Client Side Services

3.0 Services
3.0.1 Introduction
3.0.2 Role of the client and its services
3.0.3 Request for Services
3.0.4 Redirection

3.1 Print Services
3.2 Remote Services
3.3 Utility Services
3.4 Message Services
3.6 Network Services
3.6 Application Services
3.7 Database Services
3.8 Dynamic Data Exchange (DDE)
3.9 Object Linking and Embedding (OLE)
3.10 Client Tools
3.11 GUI Clients
3.12 Non-GUI Clients
3.13 OOUI (Object Oriented User Interface) Clients

3.0 Services

3.0.1 Introduction

The client in the client/server model is the desktop workstation. Any workstation that is used by a single user is a client, if multiple users share the same workstation simultaneously, then it is a server.

The famous servers like Apple Macintosh SE, an IBM PS/2 Model 30, an ALR 386/220, a Compaq System Pro, an NCD X-Terminal, Sun SPARCstation, a DEC station 6000 all are used somewhere as a client workstation. There are no specific technological characteristics of a client.
3.0.2 Role of the client and its services

- In client/server applications, functions are provided by a combination of resources using both the client workstation processor and the server processor.

- Like a database server provides data in response to an SQL request issued by the client application. The local processing by the client might calculate the invoice amount and format the response to the workstation screen.

- The client workstation can provide business functions using a mixture of personal productivity products blended with a custom application.

- The capability to cut and paste input from several different sources is one of the most powerful aspect of a client workstation.

- In the client/server model, the client is primarily a consumer of services provided by one or more server processors. Whereas the server acting a service provider responding to the client’s request.

- The client provides presentation services. The user input and final output are presentation at the client workstation. The current technology provides full support for GUI’s. The functions like field edits, context-sensitive help, navigation, training, personal data storage and manipulation frequently execute on the client workstation. All of them use the GUI and Windowing functionality.

- The client workstation can use or uses local operating system to support both basic services and the network OS interfaces.

3.0.3 Request for Services

A client workstation requests services form the attached server; whatever may be the type of processor the format of request is the same. It is the job if the NOS software to translate or add the necessary details as required by the targeted requester to the application request. OS also provides the redirection service.
3.0.4 Redirection

This service intercepts client workstation operating system calls and redirects them to the server operating system. Thus in this way the request for disk directories or disk files, printers, printer queues, serial devices, application programs and named pipes are trapped by the redirection software and redirected to the correct server location.

How does this redirection work?

Let the local disk driver be labeled A: and C: and the remote drivers labeled D:, E: and F:

1. Any request for drive A: of C: is passed through to the local file system by the redirection software. Then the request is sent to the server OS.
2. The NOS requester software constructs the remote procedure call (RPC) to include the API call to the NOS server.
3. The NOS server then processes the request as if it was executed locally and ships the response back to the application.

3.1 Print Services

- The NOS enables the client to generate print requests even when the printer is busy.
- The NOS redirector software redirects it and the print server queue manage further manage it.
- The client has facility to view the status of the print queue at any time and client is also notified when the print is completed. Similarly there are also fax services.

3.2 Remote Services

- Client invokes an application that is executed on a remote server, for example the backup services. Business functions like downloading data from host or checking list of stock process might also be invoked locally to run remotely.
- NOS provide software, which runs on the client workstation to initiate these remote applications.
§ Work from home is another concept, where the technology binds people working from home and effectively communicates with the office LAN (Which forms the Mobile Computing).

3.3 **Utility Services**

The operating system facilitates some local functions which are very often used to perform actions like edit, copy, move, compare and help which works on the client end.

3.4 **Message Services**

Messages can be sent and received to or from the network synchronously. The message services provide the buffering, scheduling and arbitration services to support this function.

3.5 **Network Services**

The client workstation communicates with the network through protocol, some set of service and API’s that creates, send and receive and format network messages. Protocols, which are supported, are NetBIOS, IPX, TCP/IP, APPC, Ethernet, Token Ring, FDDI, X.26 and SNA.

3.6 **Application Services**

In addition to the remote execution services that the NOS provide, custom applications will use their own API’s embedded in an RPC to invoke specialized services form a remote server.

3.7 **Database Services**

Database requests are made using the SQL syntax. SQL is an industry standard language supported by many vendors.

Because the language uses a standard form, the same application may be run on multiple platforms. There are syntactical differences and product extensions available from most vendors.
These are provided to improve developer productivity and system performance and should be carefully evaluated to determine whether their uses are worth the incompatibility implied by using proprietary components. Using unique features may prevent the use of another vendor's products in a larger or smaller site.

Certain extensions, such as stored procedures, are evolving into de facto standards.

The use of stored procedures is often a way of avoiding programmer use of proprietary extensions needed for performance.

A clear understanding, by the technical architects on the project, of where the standards are going is an important component of the SDE standards for the project.

### 3.8 Dynamic Data Exchange

Microsoft Windows provides several methods for transferring data between applications. DDE protocol is one of these methods. DDE protocol is a set of messages and guidelines. It sends messages between applications that share data and uses shared memory to exchange data between applications.

DDE can be used to construct hot links between applications in which data can be fed from window to window interruption and intervention.

Eg: A hot link can be created between a 3270-screen session and a word processing document. Data is linked from the 3270 window into the word processing document. Whenever the key of the data in the screen changes, the data linked into the document also changes.

DDE also supports warm links created so the server application notifies the client that the data has changed and the client can issue an explicit request to receive it. This information is attractive when the volume of changes to the server data is so great that the client prefers not to be burdened with repetitive processing.
If the server link ceases to exist at some point, use a warm link rather than hot link to ensure that the last data iteration is available. You can create request links to enable direct copy-and-paste operations between a server and client without the need for an intermediate clipboard.

DDE also facilitates the feature of extending applications. These facilities, available to the desktop user, considerably expand the opportunity for application enhancement by the user owner.

3.9 **OLE (Object Linking And Embedding)**

OLE is an extension to DDE. OLE is designed to let users to focus on data encompassing of words, numbers and graphics rather than on the software required to manipulate the data.

Here the documents are collection of objects, rather than a file, each object remembers the software that maintains it.

OLE allows users to integrate data from different applications. It allows users to share a single source of data for a particular object.

The document contains the name of the file containing the data, along with the picture of the data. When the source is updated all the documents using the data are updated as well.

Both the DDE and OLE techniques require the user to be aware of the difference between data sources. DDE and OLE provide a substantial advantage any DDE or OLE-enabled application can use any software that supports these data interchange API’s.

Not all Windows Application support OLE, to facilitate this the Microsoft has release its OLE 2.0 software development kit (SDK). This toolkit greatly simplifies OLE integration into third party, developed applications.
3.11 GUI Clients

GUI Clients: These are applications, where occasional requests to the server result from a human interacting with a GUI

(Example: Windows 3.x, NT 3.6)

3.12 Non-GUI Clients

Non-GUI Client: These are applications, generate server requests with a minimal amount of human interaction.

3.13 OOUI (Object Oriented User Interface) Clients

OOUI clients: These are applications, which are highly-iconic, Object-oriented user interface that provides seamless access to Information is in visual formats. (Example: MAC OS, Windows 96, NT 4.0)
UNIT 4
Server Side Services

4.1 Server Functionality
  4.1.1 Role of the Server
4.2 Request Processing
4.3 Print Services
4.4 Database Services
4.6 Security Services
4.6 File Services
4.7 Communication Services

4.1 Server Functionality

4.1.1 The Role of the Server
Servers provide application, file, database, print, fax, image, communications, security, systems, and network management services.

It is important to understand that a server is an architectural concept, not a physical implementation description. The same physical device can provide client and server functions.

Application servers provide business functionality to support the operation of the client workstation. In the client/server model these services can be provided for an entire or partial business function invoked through an InterProcess Communication (IPC) request for service. Either message-based requests i.e. OLTP or RPCs can be used.

A collection of application servers may work in concert to provide an entire business function. For example, in a payroll system the employee information may be managed by one application server, earnings calculated by another application server, and deductions calculated by a third application server.

These servers may run different operating systems on various hardware platforms and may use different database servers. The client application invokes these services without consideration of the technology or geographic location of the various servers.
Different Types of Servers are:

- A database engine such as Sybase, IBM, Ingres, Informix, or Oracle manages database servers.

- The file server provides the initial space, and the database engine allocates space for tables within the space provided by the file server.

- Print servers provide support to receive client documents, queue them for printing, prioritize them, and execute the specific print driver logic required for the selected printer.

- Fax servers provide support similar to that provided by print servers.

- Communications servers provide support for wide area network (WAN) communications.

- UNIX servers provide a range of product add-ons from various vendors to support the entire range of communications requirements.

- VMS servers support DECnet, TCP/IP, and SNA as well as various asynchronous and serial communications protocols.

- MVS servers provide support for SNA, TCP/IP, and some support for other asynchronous communications.

- Security at the server restricts access to software and data accessed from the server.

- Communications access is controlled from the communications server.

- A LAN administrator manages systems and network management services for the local LAN, but WAN services must be provided from some central location.
4.2 Request Processing

Steps in Request Processing are:

1. A client issues requests to the NOS services software resident on the client machine.
2. These services format the request into an appropriate RPC and issue the request to the application layer of the client protocol stack.
3. The application layer of the protocol stack on the server receives this request.

4.3 Print Services

- High-quality printers, workstation-generated faxes, and plotters are natural candidates for support from a shared server.

- The server can accept input from many clients, queue it according to the priority of the request and handle it when the device is available.

- Many organizations realize substantial savings by enabling users to generate fax output from their workstations and queue it at a fax server for transmission when the communication costs are lower.

- Incoming faxes can be queued at the server and transmitted to the appropriate client either on receipt or on request.

- In concert with workflow management techniques, images can be captured and distributed to the appropriate client workstation from the image server.

- In the client/server model, work queues are maintained at the server by a supervisor in concert with default algorithms that determine how to distribute the queued work.

- Well-defined standards for electronic document management will allow this technology to become fully integrated into the desktop work environment.
There are dramatic opportunities for cost savings and improvements in efficiency if this technology is properly implemented and used.

4.4. Database Services

Early database servers were actually file servers with a different interface.

Products such as dBASE, Clipper, FoxPro, and Paradox execute the database engine primarily on the client machine and use the file services provided by the file server for record access and free space management. These are new and more powerful implementations of the original flat-file models with extracted indexes for direct record access.

The application program, which issues lock requests and lock checks, manages currency control and by the database server, which creates a lock table that is interrogated whenever a record access lock check is generated.

Because access is at the record level, all records satisfying the primary key must be returned to the client workstation for filtering. There are no facilities to execute procedural code at the server, to execute joins, or to filter rows prior to returning them to the workstation. This lack of capability dramatically increases the likelihood of records being locked when several clients are accessing the same database and increases network traffic when many unnecessary rows are returned to the workstation only to be rejected.

The lack of server execution logic prevents these products from providing automatic partial update backout and recovery after an application, system, or hardware failure. For this reason, systems that operate in this environment require an experienced system support programmer to assist in the recovery after a failure.

Client/server database engines such as Sybase, IBM's Database Manager, Ingres, Oracle, and Informix provide support at the server to execute SQL requests issued from the client workstation. The file services are still used for space allocation and basic directory services, but all other services are provided directly by the database server.
Relational database management systems are the current technology for data management.

Relational database technology provides the current data management solution to many of the problems inherent in the flat-file and hierarchical technologies.

The primary design objective (of E.F Codd) behind SQL was to provide a data access language that could be shown mathematically to manipulate the desired data correctly.

Use of SQL

- The secondary objective was to remove any sense of the physical storage of data from the view of the user. SQL is another flat-file implementation; there are no embedded pointers. SQL uses extracted indexes to provide direct access to the rows (records) of the tables (files) of interest. Each column (field) may be used as part of the search criteria.

- SQL provides (especially with SQL2 extensions) a very powerful data access language. Its algebra provides all the necessary syntax to define, secure, and access information in an SQL database.

- The elegance of the language intrigued the user and vendor community to the extent that standards committees picked up the language and defined a set of standards around the language syntax.

- SQL1 and SQL2 define an exact syntax and a set of results for each operation. As a consequence, many software vendors have developed products that implement SQL.

- Dr. Codd has published a list of 13 rules that every SQL database engine should adhere to in order to be truly compliant.

No products today can meet all of these criteria. The criteria, however, provide a useful objective set for the standards committees and vendors to strive for. We have defined another set of product standards that we are using to evaluate SQL database engines for the development of client/server applications. In particular, products should be implemented with support for the following products and standards:
Client/Server Architecture

- ANSI SQL and IBM DB2 standards
- A variety of front-end query products
- C and COBOL SQL precompilers
- Support for and compatibility with server NOS: NetWare, OS/2 (LAN Manager, LAN Server), Windows NT, Mac System 7, and/or UNIX (VINES, SCO, Sun, HP/UX USL, SVR4_), and MVS
- Support for client Operating Systems: DOS, Windows, OS/2, Windows NT, Mac System 7, or UNIX (Solaris, USL, SCO, HP/UX, SVR4_)

Production-capable client/server database engines must be able to provide a similar operational environment to that found in the database engines present in minicomputer and mainframe computers today. Capabilities for comparison include performance, auditability, and recovery techniques. In particular, the following DBMS features must be included in the database engine:

- **Performance optimization tools**
- **Dynamic transaction backout**
- **Roll back from, roll forward to last backup**
- **Audit file recovery**
- **Automatic error detection and recovery**
- **File reclamation and repair tools**
- **Support for mirrored databases**
- **Capability to split database between physical disk drives**
- **Remote distributed database management features**
- **Maintenance of accurate and duplicate audit files on any LAN node**

In the client/server implementation, you should offload database processing to the server. Therefore, the database engine should accept SQL requests from the client and execute them totally on the server, returning only the
answer set to the client requestor. The database engine should provide support for stored procedures or triggers that run on the server.

The client/server model implies that there will be multiple concurrent user access. The database engine must be able to manage this access without requiring every developer to write well-behaved applications. The following features must be part of the database engine:

- **Locking mechanisms to guarantee data integrity**
- **Deadlock detection and prevention**
- **Multithreaded application processing**
- **User access to multiple databases on multiple servers**

### 4.6 Security Services

- Client/server applications require similar security services to those provided by host environments. Every user should be required to log in with a user ID and password.
- If passwords might become visible to unauthorized users, the security server should insist that passwords be changed regularly.
- The enterprise on the desk implies that a single logon ID and logon sequence is used to gain the authority once to access all information and process for the user has a need and right of access. Because data may be stored in a less physically secure area, the option should exist to store data in an encrypted form. A combination of the user ID and password should be required to decrypt the data.
- New options, such as floppy less workstation with integrated data encryption standard (DES) coprocessors, are available from vendors such as Beaver Computer Company. These products automatically encrypt or decrypt data written or read to disk or a communication line. The encryption and decryption are done using the DES algorithm and the user password. This ensures that no unauthorized user can access stored data or communications data.
- This type of security is particularly useful for laptop computers participating in client/server applications, because laptops do not operate in surroundings with the same physical security of an office.
4.6 File Services

File services handle access to the virtual directories and files located on the client workstation and to the server's permanent storage. These services are provided through the redirection software implemented as part of the client workstation operating environment.

The file services provide this support at the remote server processor. In the typical implementation, software, shared data, databases, and backups are stored on disk, tape, and optical storage devices that are managed by the file server.

To minimize the effort and effect of installation and maintenance of software, software should be loaded from the server for execution on the client.

New versions can be updated on the server and made immediately available to all users.

In addition, installation in a central location reduces the effort required for each workstation user to handle the installation process. Because each client workstation user uses the same installation of the software, optional parameters are consistent, and remote help desk operators are aware of them.

Backups of the server can be scheduled and monitored by a trained support person. Backups of client workstations can be scheduled from the server, and data can be stored at the server to facilitate recovery. Tape or optical backup units are typically used for backup; these devices can readily provide support for many users.

A support person who ensures that the backup functions are completed readily monitors a central location. With more organizations looking at multimedia and image technology, large optical storage devices are most appropriately implemented as shared servers.
4.7 Communications Services

- Client/server applications require LAN and WAN communication services.
- Basic LAN services are integral to the NOS.
- Various communications server products provide WAN services.
UNIT 5
Network and Protocol Basics

5.0 Communication Network
5.1 Local Area Network
5.2 Metropolitan Area Network
5.3 Wide Area Network
5.4 Network Structure
5.6 OSI Model
5.6 TCP/IP Architecture
5.7 TCP/IP Protocols

5.0 Communication Network

5.0.1 Introduction
In the world of computers, networking is the practice of linking two or more computing devices together for the purpose of sharing data. Networks are built with a combination of computer hardware and computer software. Networks can be categorized in several different ways. One method defines the type of a network according to the geographic area it spans. Alternatively, networks can also be classified based on topology or on the types of protocols they support.

5.0.2 Introduction to Network Type
One way to categorize the different types of computer network designs is by their scope or scale. For historical reasons, the networking industry refers to nearly every type of design as some kind of area network. Types of networks are: LAN - Local Area Network

- WLAN - Wireless Local Area Network
- LAN - Wide Area Network
- MAN - Metropolitan Area Network
• **SAN** - *Storage Area Network, System Area Network, Server Area Network, or sometimes Small Area Network*
• **CAN** - *Campus Area Network, Controller Area Network, or sometimes Cluster Area Network*
• **PAN** - *Personal Area Network*
• **DAN** – *Desk Area Network*

LAN and WAN were the original categories of area networks.

### 5.1 LAN - Local Area Network

A **LAN** connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs, and occasionally a LAN will span a group of nearby buildings. In TCP/IP networking, a LAN is often but not always implemented as a single IP subnet. A LAN in turn often connects to other LANs, and to the Internet or other **WAN**.

Most local area networks are built with relatively inexpensive hardware such as **Ethernet** cables, network adapters, and hubs. Wireless **LAN** and other more advanced **LAN** hardware options also exist.

The most common type of local area network is an **Ethernet** **LAN**. The smallest home **LAN** can have exactly two computers; a large **LAN** can accommodate many thousands of computers.

### 5.2 Metropolitan Area Network

**MAN** is a network spanning a physical area larger than a **LAN** but smaller than a **WAN**, such as a city. A **MAN** is typically owned and operated by a single entity such as a government body or large corporation.

### 5.3 WAN - Wide Area Network

**WAN** spans a large physical distance. The **Internet** is the largest **WAN**, spanning the Earth.
A WAN is a geographically dispersed collection of LANs. A network device called a router connects LANs to a WAN. In IP networking, the router maintains both a LAN address and a WAN address.

A WAN differs from a LAN in several important ways. Most WANs (like the Internet) are not owned by any one organization but rather exist under collective or distributed ownership and management.

WANs tend to use technology like ATM, Frame Relay and X.26 for connectivity over the longer distances.

**Other Types of Area Networks**

While LAN and WAN are by far the most popular network types mentioned, you may also commonly see references to these others:

- **Wireless Local Area Network** - a LAN based on WiFi wireless network technology
- **Campus Area Network** - a network spanning multiple LANs but smaller than a MAN, such as on a university or local business campus.
- **Storage Area Network** - connects servers to data storage devices through a technology like Fiber Channel.
- **System Area Network** - links high-performance computers with high-speed connections in a cluster configuration. Also known as Cluster Area Network

**5.4 Network structure**

The Network Structure describes the method of how data on a network is organized and viewed.

Networks are usually classified using three properties:

i. Topology
ii. Protocol

iii. Architecture

5.4.1 Topology

Topology specifies the geometric arrangement of the network. Common topologies are a bus, ring, and star.

Bus Topology:

A bus topology means that each computer on the network is attached to a common central cable, called a bus or backbone. This is a rather simple network to set up.

Ethertnets use this topology.

Ring Topology:

A ring topology means that each computer is connected to two others, and they arranged in a ring shape. These are difficult to set up, but offer high bandwidth.

Star Topology:

A star topology means all computers on the network are connected to a central hub. These are easy to set up, but bottlenecks can occur because all data must pass through the hub. The Figure 5.4.a below depicts these three topologies:
5.4.2 Protocols

Protocol specifies a common set of rules and signals the computers on the network use to communicate. There are many protocols, each having advantages over others. The most popular ones are:

- **TCP/IP**:
  - Transmission Control Protocol / Internet Protocol. This was originally developed by the Defense Department of the US to allow dissimilar computers to talk.
  - Today, as many of us know, this protocol is used as the basis for the internet. Because it must span such large distances and cross multiple, smaller networks, TCP/IP is a routable protocol, meaning it can send data through a router on its way to its destination. In the long run, this slows things down a little, but this ability makes it very flexible for large networks.
• **IPX/SPX:**
  - Developed by Novell for use with its NetWare NOS, but Microsoft built compatibility into both NT and Windows 9x. IPX is like an optimized TCP/IP.
  - It, too, is a routable protocol, making it handy for large networks, but it allows quicker access over the network than TCP/IP. The downfall is that it doesn’t work well over analog phone lines.
  - IPX continually checks the status of transmission to be sure all the data arrives. This requires extra bandwidth, where analog phone lines don’t have much to begin with. This results in slow access. Of course, the data is more reliable with IPX.

• **NetBEUI:**
  - Designed for small LANs, this protocol developed by Microsoft is quite fast.
  - It lacks the addressing overhead of TCP/IP and IPX, which means it can only be used on LANs.
  - You cannot access networks via a router.

### 5.4.3 Architecture

Architecture refers to one of the two major types of network architecture: Peer-to-peer or client/server.

In a **Peer-to-Peer** networking configuration, there is no server, and computers simply connect with each other in a workgroup to share files, printers, and Internet access.

This is most commonly found in home configurations, and is only practical for workgroups of a dozen or less computers.

In a **client/server** network, there is usually an NT Domain Controller, which all of the computers log on to. This server can provide various services,
including centrally routed Internet Access, mail (including e-mail), file sharing, and printer access, as well as ensuring security across the network.

This is most commonly found in corporate configurations, where network security is essential.

5.6 OSI Model (Open Systems Interconnect)

The OSI reference model shown in Figure 5.6 provides an industry standard framework for network and system interoperability.

The existence of heterogeneous LAN environments in large organizations makes interoperability a practical reality. Organizations need and expect to view their various workgroup LANs as an integrated corporate-wide network.

Citicorp, for example, is working to integrate its 100 independent networks into a single global net.

1. The OSI model provides the framework definition for developers attempting to create interoperable products.

2. Because many products are not yet OSI-compliant, there often is no direct correspondence between the OSI model and reality.

The OSI model defines seven protocol layers and specifies that each layer be insulated from the other by a well-defined interface.

The following fig 5.6 depicts the OSI Model with 7 layers.

<table>
<thead>
<tr>
<th>Application Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation Layer</td>
</tr>
<tr>
<td>Session Layer</td>
</tr>
<tr>
<td>Transport Layer</td>
</tr>
<tr>
<td>Network Layer</td>
</tr>
<tr>
<td>Data Link Layer</td>
</tr>
<tr>
<td>Physical Layer</td>
</tr>
</tbody>
</table>

Fig 5.6 OSI Model
1. Physical Layer

- The physical layer is the lowest level of the OSI model and defines the physical and electrical characteristics of the connections that make up the network.
- It includes such things as interface specifications as well as detailed specifications for the use of twisted-pair, fiber-optic, and coaxial cables.
- Standards of interest at this layer for client/server applications are IEEE 802.3 (Ethernet), and IEEE 802.6 (Token Ring) that define the requirements for the network interface card (NIC) and the software requirements for the media access control (MAC) layer.
- Other standards here include the serial interfaces EIA232 and X.21.

2. Data Link Layer

- The data link layer defines the basic packets of data expected to enter or leave the physical network. Bit patterns, encoding methods, and tokens are known to this layer.
- The data link layer detects errors and corrects them by requesting retransmission of corrupted packets or messages.
- This layer is actually divided into two sublayers:
  - The media access control (MAC)
    - The MAC sublayer has network access responsibility for token passing, collision sensing, and network control. The LLC sublayer operates above the MAC and sends and receives data packets and messages.
  - The logical link control (LLC).
- Ethernet, Token Ring, and FDDI define the record format of the packets (frames) being communicated between the MAC layer and Network layer. The internal formats are different and without conversion workstations cannot interoperate with workstations that operate with another definition.
3. **Network Layer**

- The network layer is responsible for switching and routing messages to their proper destinations. It coordinates the means for addressing and delivering messages.
- It provides for each system a unique network address, determines a route to transmit data to its destination, segments large blocks of data into smaller packets of data, and performs flow control.

4. **Transport Layer**

- When a message contains more than one packet, the transport layer sequences the message packets and regulates inbound traffic flow. The transport layer is responsible for ensuring end-to-end error-free transmission of data.
- The transport layer maintains its own addresses that get mapped onto network addresses. Because the transport layer services process on systems, multiple transport addresses (origins or destination) can share a single network address.

5. **Session Layer**

- The session layer provides the services that enable applications running at two processors to coordinate their communication into a single session.
- A session is an exchange of messages—a dialog between two processors.
- This layer helps create the session, inform one workstation if the other drops out of the session, and terminate the session on request.

6. **Presentation Layer**

- The presentation layer is responsible for translating data from the internal machine form of one processor in the session to that of the other.
7. Application Layer

- The application layer is the layer to which the application on the processor directly talks. The programmer codes to an API defined at this layer.
- Messages enter the OSI protocol stack at this level, travel through the layers to the physical layer, across the network to the physical layer of the other processor, and up through the layers into the other processor application layer and program.

5.6 TCP/IP Architecture

History

Many organizations were unable to wait for the completion of the OSI middle-layer protocols during the 1980s. Vendors and users adopted the Transmission Control Protocol/Internet Protocol (TCP/IP), which was developed for the United States military Defense Advanced Research Projects Agency (DARPA) ARPANET networks. ARPANET was one of the first layered communications networks and established the precedent for successful implementation of technology isolation between functional components.

Today, the Internet is a worldwide-interconnected network of universities, research, and commercial establishments; it supports thirty million US users and fifty million worldwide users. Additional networks are connected to the Internet every hour of the day. In fact growth is now estimated at 16 percent per month. The momentum behind the Internet is tremendous.

The TCP/IP protocol suite is now being used in many commercial applications. It is particularly evident in internetworking between different LAN environments.

The TCP/IP

TCP/IP is specifically designed to handle communications through "networks of interconnected networks."

In fact, it has now become the de facto protocol for LAN-based Client/Server connectivity and is supported on virtually every computing platform.
More importantly, most interprocess communications and development tools embed support for TCP/IP where multiplatform interoperability is required.

It is worth noting that IBM has followed this growth and not only provides support for TCP/IP on all its platforms, but now enables the transport of its own interoperability interfaces (such as CPIC, APPC) on TCP/IP.

**TCP/IP's Architecture**

The TCP/IP protocol suite is composed of the following components: a network protocol (IP) and its routing logic, three transport protocols (TCP, UDP, and ICMP), and a series of session, presentation and application services. The following sections highlight those of interest.

### 5.7 TCP/IP Protocols

#### 1. Internet Protocol

IP represents the network layer and is equivalent to OSI's IP or X.26. A unique network address is assigned to every system, whether the system is connected to a LAN or a WAN. The system comes with its associated routing protocols and lower level functions such as network-to-physical address resolution protocols (ARP). Commonly used routing protocols include RIP, OSPF, IGRP, and Cisco's proprietary protocol. The community to be the standards-based preferred protocol for large networks has adopted OSPF.

#### 2. Transport Protocols

TCP provides Transport services over IP. It is connection-oriented, meaning it requires a session to be set up between two parties to provide its services. It ensures end-to-end data transmission, error recovery, ordering of data, and flow control. TCP provides the kind of communications that users and programs expect to have in locally connected sessions.

UDP provides connectionless transport services, and is used in very specific applications that do not require end-to-end reliability such as that provided by TCP.
3. Telnet

Telnet is an application service that uses TCP. It provides terminal emulation services and supports terminal-to-host connections over an internetwork. It is composed of two different portions: a client entity that provides services to access hosts and a server portion that provides services to be accessed by clients. Even workstation operating systems such as OS/2 and Windows can provide telnet server support, thus enabling a remote user to log onto the workstation using this method.

4. File Transfer Protocol (FTP)

FTP uses TCP services to provide file transfer services to applications. FTP includes a client and server portion. Server FTP listens for a session initiation request from client FTP. Files may be transferred in either direction, or ASCII and binary file transfer is supported. FTP provides a simple means to perform software distribution to hosts, servers, and workstations.

5. Simple Network Management Protocol (SNMP)

SNMP provides intelligence and services to effectively manage an internetwork. It has been widely adopted by hub, bridge, and router manufacturers as the preferred technology to monitor and manage their devices.

SNMP uses UDP to support communications between agents—intelligent software that runs in the devices—and the manager, which runs in the management workstation. Two basic forms of communications can occur: SNMP polling (in which the manager periodically asks the agent to provide status and performance data) and trap generation (in which the agent proactively notifies the manager that a change of status or an anomaly is occurring).

6. Network File System (NFS)

The NFS protocol enables the use of IP by servers to share disk space and files the same way a Novell or LAN Manager network server does. It is useful in environments in which servers are running different operating systems. However, it does not offer support for the same administration facilities that a NetWare environment typically provides.
7. **Simple Mail Transfer Protocol (SMTP)**

SMTP uses TCP connections to transfer text-oriented electronic mail among users on the same host or among hosts over the network. Developments are under way to adopt a standard to add multimedia capabilities (MIME) to SMTP. Its use is widespread on the Internet, where it enables any user to reach millions of users in universities, vendor organizations, standards bodies, and so on. Most electronic mail systems today provide some form of SMTP gateway to let users benefit from this overall connectivity.

**TCP/IP and Internetworks**

Interestingly, the interconnected LAN environment exhibits many of the same characteristics found in the environment for which TCP/IP was designed. In particular

- **Routing:** Internetworks need support for routing; routing is very efficient in TCP/IP environments with efficient protocols such as OSPF.

- **Connections versus Connectionless:** LAN activity includes both; the TCP/IP protocol suite efficiently supports both within an integrated framework.

- **Administrative Load Sensitivity:** A LAN administrative support is usually limited; contrary to IBM's SNA, TCP/IP environments contain a tremendous amount of dynamic capabilities, in which devices and networks are dynamically discovered, and routing tables are automatically maintained and synchronized.

- **Networks of Networks:** TCP/IP provides extreme flexibility as the administrative approach to the management of federations of networks. Taking advantage of its dynamic nature, it enables very independent management of parts of a network (if appropriate).
UNIT 6
Hardware

6.1 Introduction

6.1.1 The Hardware Issues:

6.1.2 Connectivity

6.1.3 Communication Server

6.2 Hardware/Network Acquisition

6.3 Different types of Computing Devices

6.3.1 PC-Level Processing Units

6.3.2 Macintosh

6.3.3 Notebooks

6.3.4 Pen

6.3.5 UNIX Workstation

6.4 Server Hardware

6.5 Data Storage

6.5.1 Magnetic Disk

6.5.2 Mirrored Disk

6.5.3 RAID-Disk Array

6.5.4 Tape

6.5.5 Optical Disks

6.5.6 CD-ROM

6.5.7 WORM

6.5.8 Erasable Optical

6.5.9 Erasable Optical

6.5.10 Network Interface Cards (NICs)

6.5.11 Power Protection Devices

6.5.12 Uninterruptible Power Supply (UPS)

6.1 Introduction
6.1.1 The Hardware Issues:

We already know by now that the "Advantages of Client/Server Computing," the cost of powerful hardware for client/server computing has declined dramatically in the last few years. Nevertheless, this power must be packaged properly, and cost still must be considered in the design and purchasing decision.
Hardware that provides the client/server, LAN-to-LAN, and LAN-to-WAN connectivity must be acquired for clients, servers, data storage, and the networks.

Entry-level client workstations can range from a basic Intel-based PC to an entry-level Apple Macintosh or an X-Terminal. These entry-level clients start at about $1,000 and use LAN servers for printing, backup, software storage, application execution, and WAN connectivity.

Whereas the High-end client workstations can cost more than $60,000 for engineering stations that provide advanced capabilities such as a gigabyte or more of local storage, high-resolution graphics monitors, 100-MIPS processing, direct WAN connectivity, 1000-dpi color printing, or professional multimedia development tools.

The average client workstation has dropped from $6000 to $2000 in the last two years. This buys a configuration with the processing power equivalent to an 8Mbyte Intel 33-MHz 486DX PC with local storage of 260Mbytes, LAN connectivity, and a VGA-equivalent monitor.

This cost level is not expected to decline much further, because GUI software and reengineered application requirements will steadily increase the processing power requirements for entry-level machines.

Many organizations also have client/server applications that use the services of existing IBM 370 mainframes running VM, MVS, or VSE, DEC VAX minicomputers running VMS or Ultrix, and large RISC-based systems running UNIX—all as high-end servers.

Other mainframe and minicomputer hardware platforms, running proprietary operating systems, are frequently used in terminal emulation mode from the client workstation.

Servers based on the IBM, DEC, and UNIX operating systems will provide application services using existing applications through terminal emulation or RPC-invoked application services. These same servers will provide connectivity and database services to the first client/server applications in an organization.
6.1.2 Connectivity

- Connectivity requires every client workstation to be connected to a LAN or through a WAN to a remote server. In the usual situation, the workstation is connected through an Ethernet, Token Ring, FDDI, CDDI, or occasionally a parallel or serial interface to the LAN.
- The primary connection types require a network interface card (NIC) to be inserted in the workstation to provide the protocol processing necessary to establish and maintain the connection. The cost of LAN connectivity has declined rapidly in parallel with the industry reduction in workstation costs.
- Cabling costs vary widely, depending on the physical difficulty of installation and whether the network planners choose unshielded twisted-pair (UTP), shielded twisted-pair (STP), or glass-fiber cables. Glass-fiber termination equipment is more costly than twisted-pair, although the costs are declining.
- Today, many vendors provide the hardware for these connections. Each vendor offers some advantages in terms of cost, performance, and reliability.

6.1.3 Communication Server

- WAN connectivity requires each workstation to be directly connected to the WAN or to a communications server connected to the WAN.
- Most new LANs are installed using communications servers. There are cost, performance, and especially network management reasons for using a LAN communications server.
- A substantial advantage accrues because there is no need to cable each workstation to the WAN. Workstations that are individually connected to the WAN require an embedded controller card for synchronous communications and either a modem or serial connection for asynchronous communications.
- A major advantage of the communications server is its ability to multiplex a high-speed communications line and provide bandwidth on demand to each client workstation. Only the single LAN cable and LAN controller are needed to provide workstation connectivity in the server implementation.
• Data storage can be provided to a client from a local disk or through the file services of the NOS. Local disk storage requires the workstation to have its own disk devices. Server storage involves large shared server disks. In either case, a backup capability must be provided. This can be done through local diskette or tape devices or through a server tape, disk, or optical device.

6.2 Hardware/Network Acquisition

• Before selecting client hardware for end users, most organizations should define standards for classes of users.
• This set of standards simplifies the selection of the appropriate client hardware for a user and allows buyers to arrange purchasing agreements to gain volume pricing discounts.
• There are a number of issues to consider when selecting the client workstation, including processor type, coprocessor capability, internal bus structure, size of the base unit, and so on.
• However, of these issues, one of the most overlooked regarding client/server applications is the use of a GUI.
• GUI applications require VGA or better screen drivers. Screens, larger than the 16-inch PC standard, are required for users who normally display several active windows at one time; the more windows active on-screen, the larger the monitor viewing area requirements.
• The use of image, graphics, or full-motion video requires a large screen with very high resolution for regular usage.
• It is important to remember that productivity is dramatically affected by inability to easily read the screen all day. Inappropriate resolution will lead to fatigue and inefficiency.
• The enterprise on the desk requires that adequate bandwidth be available to provide responsiveness to the desktop user. If regular access to off LAN data is required, a router based internetworking implementation will be required.
• If only occasional off LAN access is required, bridges can be used. Routers provide the additional advantage of support for multiprotocol internetworking.
• This is frequently necessary as organizations install 10BaseT Ethernet into an existing Token Ring environment. Fast Ethernet and FDDI are becoming more prevalent as multimedia applications are delivered.
6.3 Different types of Computing Devices

6.3.1 PC-Level Processing Units

- Client/server applications vary considerably in their client processing requirements and their I/O demands on the client processor and server.
- In general, clients that support protected-mode addressing should be purchased. This implies the use of 32-bit processors—perhaps with a 16-bit I/O bus if the I/O requirement is low.
- Low means the client isn't required to send and receive large amounts of data, such as images, which could be 100K bytes or larger, on a constant basis.
- Windowed applications require considerable processing power to provide adequate response levels.
- The introduction of application integration via DCE, OLE, and DOE significantly increases the processing requirements at the desktop.
- The recommended minimum configuration for desktop processors has the processing capacity of a 33Mhz Intel 486SX.

6.3.2 Macintosh

- The Mac System 7 operating system is visually intuitive and provides the best productivity when response time to GUI operations is secondary.
- The Motorola 68040, 8Mbytes RAM, 120Mbyte disk is recommended.
- By early 1996, the availability of PowerPC technology and the integration of System 7 with AIX and Windows means that users will need considerably more processor capacity.
- Fortunately, the PowerPC will provide this for the same or lower cost than the existing Motorola technology.

6.3.3 Notebooks

- The notebook computer is the fastest growing market today. Users working remotely on a regular basis may find that a notebook computer best satisfies their requirements.
• The current technology in this area is available for Intel PC, Apple Macintosh, and SPARC UNIX processors.
• Relatively slower speed of disk I/O on notebooks makes it preferable to install extra RAM, creating "virtual" disk drives.
• A minimal configuration is a processor with the equivalent processing power of a 33 MHz Intel 486SX, 8mbytes of RAM and 140Mbytes of disk.
• In addition, the notebook with battery should weigh less than seven pounds and have a battery life of three hours.
• If the application will run a remote GUI, it is desirable to install software to compress the GUI and V.32 modem communications at 9600 bps or V.32bis at 14400 bps, employing V.42 and V.42bis compression, respectively.
• The introduction of PCMCIA technology, credit card size modems, and flash memory are available to upgrade the notebook.

**6.3.4 Pen**

• Pen-based clients provide the capability to operate applications using a pen to point and select or write without the need for a mouse or keyboard.
• Frequently, they are used for verification, selection, and inquiry applications where selection lists are available. Developers using this technology use object-oriented software techniques that are RAM-intensive.
• The introduction of personal digital assistant (PDA) technology in 1993 has opened the market to pocket size computing. During 1994, this technology will mature with increased storage capacity through cheaper, denser RAM and flash memory technology. The screen resolution will improve, and applications will be developed that are not dependent upon cursive writing recognition.
• In combination with wireless technology advances, this will become the personal information source for electronic news, magazines, books, and so on.

**6.3.5 UNIX Workstation**

• UNIX client workstations are used when the client processing needs are intensive. In many applications requiring UNIX, X-terminals connected to a UNIX presentation server will be the clients of choice.
• A UNIX client workstation will then have more processing power than a PC client.
• The introduction of software from SunSoft, Insignia Solutions, and Locus Computing that supports the execution of DOS and Windows 3.x applications in a UNIX window makes the UNIX desktop available to users requiring software from both environments.

6.4 Server Hardware

• Server requirements vary according to the complexity of the application and the distribution of work. Because servers are multiuser devices, the number of active users is also a major sizing factor.
• Servers that provide for 32-bit preemptive multitasking operating systems with storage protection are preferred in the multiuser environment.
• Intel-based tower PCs and Symmetric Multi-Processors (SMPs) are commonly used for workgroup LANs with file and application service requirements.
• Most PC vendors provide a 66 MHz Intel 486DX or Intel Pentium for this market in 1994. Vendors such as IBM, Compaq, and NetFrame provide SMP products.
• Traditional UNIX vendors, such as Sun, HP, IBM, and Pyramid provide server hardware for applications requiring UNIX stability and capacity for database and application servers and large workgroup file services.
• The SMP products, in conjunction with RAID disk technology, can be configured to provide mainframe level reliability for client/server applications.
• It is critical that the server be architected as part of the systems management support strategy to achieve this reliability.

6.5 Data Storage

Permanent storage requirements are very application-specific. In addition to quantity of disk storage, the issues of performance and reliability must be considered.
6.5.1 Magnetic Disk

Disk storage devices should use the SCSI-2 standard controller interface. This provides the best performance in a standards-based environment.

Many vendors provide high-capacity, high-performance, and highly reliable disk devices for this controller.

The use of high-capacity cache storage dramatically improves performance. Most current SCSI-2 controllers are configurable with 266K or more cache. This is an important, yet frequently overlooked component of the architecture.

New drives are available in the traditional 3.6 sizes with 1.0-1.6Gbyte capacity.

The use of compression software can easily double this capacity. With the increasing size of GUI software and the introduction of multimedia applications, the demand for disk capacity will increase rapidly during 1994 and beyond.

6.5.2 Mirrored Disk

When applications require high reliability, it may be appropriate to use a configuration that supports mirrored disks. With this configuration, data is automatically written to two disks. This enables the application to continue even if a failure occurs on one disk.

System files and data files should be considered for mirroring. Even though system files are usually read-only, the number of users affected by unavailability of the files may justify this redundancy.

In addition, performance can improve since dual reads can be handled in parallel.

6.5.3 RAID-Disk Array

Traditional magnetic disk technology is often referred to as single large expensive disk (SLED). Very high performance and high availability can be achieved through a redundant array of inexpensive drives (RAID).
These enable data files to be spread across several physical drives. Data also can be mirrored as part of this configuration.

RAID technology provides a considerable performance advantage because many parallel I/O operations can be processed at the same time.

High capacity caches must be used in conjunction with RAID to achieve optimum performance. The size will be identified as part of the architecture definition.

### 6.5.4 Tape

Although most permanently stored data uses disk, tape is a very popular form of low-cost magnetic storage and is used primarily for backup purposes.

The standard backup tape device today is digital audiotape (DAT). These tapes provide approximately 1.2 Gigabytes of storage on a standard cartridge-size cassette tape.

Tape is a sequential medium and does not adequately support direct (random) access to information.

If an organization standardizes on a single tape format and technology, distribution of information by mailing tapes can be a cost-effective communications mechanism for large quantities of information that do not require real-time transmission or accessibility.

### 6.5.5 Optical Disks

Optical disk storage technology provides the advantage of high-volume, economical storage with somewhat slower access times than traditional magnetic disk storage.

### 6.5.6 CD-ROM

Compact disk-read only memory (CD-ROM) optical drives are used for storage of information that is distributed for read-only use. A single CD-ROM can hold up to 800MB of information.
Software and large reports distributed to a large number of users are good candidates for this media. CD-ROM also is more reliable for shipping and distribution than magnetic disks or tapes.

The advent of multimedia applications and the resulting storage requirements will further drive the demand for CD-ROM.

In 1993, the speed of CD-ROM technology was doubled through a doubling of the rotation of the drive. Newer drives will triple-spin and quad-spin. The speed of the drive is very critical for applications that use the CD-ROM interactively.

The addition of large cache SCSI-2 controllers can also significantly improve performance. The architecture definition must look at the business requirement in determining the appropriate configuration. Poor selection will result in unacceptable performance, excessive cost, or both.

**6.5.7 WORM**

Write once, read many (WORM) optical drives are used to store information that is to be written to disk just once but read many times. This type of storage is frequently used to archive data that should not be modified.

Traffic tickets issued by police departments are scanned and stored on WORM drives for reference on payment or nonpayment.

The WORM technology guarantees that the image cannot be tampered with. A magnetic drive can be used to store an index into the data on the WORM drive.

Data can be effectively erased from a WORM by removing reference to it from the index. This can be useful when a permanent audit trail of changes is required.

**6.5.8 Erasable Optical**

Erasable optical drives are used as an alternative to standard magnetic disk drives when speed of access is not important and the volume of data stored is large.
Client/Server Architecture

They can be used for image, multimedia, backup, or high-volume, low-activity storage.

6.5.9 Network Interface Cards (NICs)

Client and server processors are attached to the LAN through NICs. These provide the physical connectivity to the wire and the protocol support to send/receive messages. The most popular network protocols today are Token Ring, Ethernet, and FDDI.

6.5.10 Power Protection Devices

A lot has been written in books, magazines, and journals about computer hardware and software; and a number of computer specialty businesses are dedicated to helping you work through issues of specific concern to your business objectives.

6.5.11 Uninterruptible Power Supply (UPS)

Prices for UPS have declined to the point where they are widely used for LAN server protection. These units contain battery backups to enable at least a graceful power-down sequence.

All buffers can be purged and files closed so that no information is lost. Other units provide 16-90 minutes of power backup to handle most power failure situations.

6.5.12 Surge Protectors

The electronics in computer systems are affected by power fluctuations. Protection can be obtained through the use of surge protection equipment. Every client and server processor and all peripherals should be wired through a surge protector. Most UPS systems include integrated surge protectors.
Unit 7
Application development management issues

7.0 Need to Improve Technology Professional Productivity
7.0.1 Need for Platform Migration and Reengineering of Existing Systems
7.0.2 Need for a Common Interface Across Platforms
7.0.3 Increase in Applications Development by Users

7.2 Client/Server Systems Development Methodology
7.3 Project Management
7.4 Architecture Definition
7.5 Systems Development Environment (SDE)
7.6 CASE
7.7 Object-Oriented Programming (OOP) and CASE Tools

7.0 Need to Improve Technology Professionals' Productivity

The Index Group reports that the Computer-Aided Software Development (CASE) and other technologies that speed software development are cited by 70 percent of the top IT executives surveyed as the most critical technologies to implement.

This new breed of software tools helps organizations respond more quickly by cutting the time it takes to create new applications and making them simpler to modify or maintain. Old methods, blindly automating existing manual procedures, can hasten a company's death knell.

Companies need new, innovative mission-critical systems to be built quickly, with a highly productive, committed professional staff partnered with end-users during the requirements, design, and construction phases.

The client/server development model provides the means to develop horizontal prototypes of an application as it is designed. The user will be encouraged to think carefully about the implications of design elements. The visual presentation through the workstation is much more real than the paper representation of traditional methods.
**Yourdon** reports that less than 20 percent of development shops in North America have a methodology of any kind, and even a lower percentage actually use the methodology.

Input Research reports that internally developed systems are delivered on time and within budget about 1 percent of the time. They compare this result to those outsourced through systems integration professionals who use high-productivity environments, which are delivered on time and within budget about 66 percent of the time.

The use of a proven, formal methodology significantly increases the likelihood of building systems that satisfy the business need and are completed within their budgets and schedules.

Yourdon estimates that 60 percent of errors in a final system and 76 percent of the cost of error removal can be traced back to errors in the analysis phase.

CASE tools and development methodologies that define systems requirements iteratively with high and early user involvement have been proven to significantly reduce analysis phase errors.

### 7.0.1 Need for Platform Migration and Reengineering of Existing Systems

Older and existing applications are being rigorously reevaluated and in some cases terminated when they don't pay off. A 16-percent drop in proprietary technology expenditures was measured in 1993 and this trend will continue as organizations move to open systems and workstation technology.

BPR attempts to reduce business process cost and complexity by moving decision-making responsibility to those individuals who first encounter the customer or problem. Organizations are using the client/server to bring information to the workplace of empowered employees.

The life of an application tends to be 6 to 16 years, whereas the life of a technology is much shorter—usually one to three years.

Tremendous advances can be made by reengineering existing applications and preserving the rule base refined over the years while taking advantage of the orders-of-magnitude improvements that can be achieved using new technologies.
7.0.2 Need for a Common Interface Across Platforms

Graphical user interfaces (GUIs) that permit a similar look and feel and front-end applications that integrate disparate applications are on the rise.

A 1991 *Information Week* survey of 167 IT executives revealed that ease of use through a common user interface across all platforms is twice as important as the next most important criteria as a purchasing criterion for software. This is the single-system image concept.

Of prime importance to the single-system image concept is that every user from every workstation have access to every application for which they have a need and right without regard to or awareness of the technology.

Developers should be equally removed from and unconcerned with these components. Development tools and APIs isolate the platform specifics from the developer. When the single-systems image is provided, it is possible to treat the underlying technology platforms as a commodity to be acquired on the basis of price-performance without concern for specific compatibility with the existing application.

Hardware, operating systems, and database engines, communication protocols—all these must be invisible to the application developer.

7.0.3 Increase in Applications Development by Users

As workstation power grows and dollars-per-MIPS fall, more power is moving into the hands of the end user. The Index Group reports that end users are now doing more than one-third of application development; IT departments are functioning more like a utility. This is the result of IT department staff feeling the squeeze of maintenance projects that prevent programmers from meeting critical backlog demand for new development.

This trend toward application development by end-users will create disasters without a consistent, disciplined approach that makes the developer insensitive to the underlying components of the technology.

End-user application developers also must understand the intricacies of languages and interfaces.
Object-oriented technologies embedded in SDE have regularly demonstrated to produce new development productivity gains of 2 to 1 and maintenance productivity improvements of 6 to 1 over traditional methods—for example, process-driven or data-driven design and development.

More recently mature OO SDEs with a strong focus on object reusability are achieving productivity gains of 10 to 1 over traditional techniques.

Production-capable technologies are now available to support the development of client/server applications. The temptation and normal practice is to have technical staff read the trade press and select the best products from each category, assuming that they will combine to provide the necessary development environment. In fact, this almost never works. When products are not selected with a view as to how they will work together, they do not work together.

Thus, the best Online Transaction Processing (OLTP) package may not support YOUR best database. Your security requirements may not be met by any of your tools. Your applications perform well, but it may take forever to change them. Organizations must architect an environment that takes into account their particular priorities and the suite of products being selected. The selection of tools will provide the opportunity to be successful.

An enterprise-wide architecture strategy must be created to define the business vision and determine a transformation strategy to move from the current situation to the vision. This requires a clear understanding of industry standards, trends, and vendor priorities.

Combining the particular business requirements with industry direction it is possible to develop a clear strategy to use technology to enable the business change. Without this architecture strategy, decisions will be made in a vacuum with little business input and usually little clear insight into technology direction.

The next and necessary step is to determine how the tools will be used within your organization. This step involves the creation of your SDE. Without the integration of an SDE methodology, organizations will be unable to achieve the benefits of client/server computing. Discipline and standards are essential to create platform-independent systems. With the uncertainty over which technologies will survive as standards, the isolation
of applications from their computing platforms is an essential insurance policy.

7.2 Client/Server Systems Development Methodology

The purpose of a methodology is to describe a disciplined process through which technology can be applied to achieve the business objectives. Methodology should describe the processes involved through the entire life cycle, from BPR and systems planning through and including maintenance of systems in production.

Most major systems integrators and many large in-house MIS groups have their own life cycle management methodology. Andersen Consulting, for example, has its Foundation, BSG has its Blueprint, and SHL Systemhouse has its own SHL Transform—the list goes on and on.

These companies offer methodologies tuned for the client/server computing environment. However, every methodology has its own strengths, which are important to understand as part of the systems integration vendor selection process.

The following table depicts the details of the major activities of each stage of the systems integration life cycle methodology.

No activity is complete without the production of a formal deliverable that documents, for user signoff, the understanding gained at that stage.

The last deliverable from each stage is the plan for the next stage.
<table>
<thead>
<tr>
<th><strong>SILC Phase</strong></th>
<th><strong>Major Activities</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Systems Planning</strong></td>
<td>Initiate systems planning</td>
</tr>
<tr>
<td>Gather data</td>
<td>Identify current situation</td>
</tr>
<tr>
<td>Describe existing systems</td>
<td>Define requirements</td>
</tr>
<tr>
<td>Analyze applications and data architectures</td>
<td>Analyze technology platforms</td>
</tr>
<tr>
<td>Prepare implementation plan</td>
<td></td>
</tr>
<tr>
<td><strong>Project Initiation</strong></td>
<td>Screen request</td>
</tr>
<tr>
<td>Identify relationship to long-range systems plan</td>
<td>Initiate project</td>
</tr>
<tr>
<td>Prepare plan for next phase</td>
<td></td>
</tr>
<tr>
<td><strong>Architecture Definition</strong></td>
<td>Gather data</td>
</tr>
<tr>
<td>Expand the requirements to the next level of detail</td>
<td>Conceptualize alternative solutions</td>
</tr>
<tr>
<td>Develop proposed conceptual architecture</td>
<td>Select specific products and vendors</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>Gather data</td>
</tr>
<tr>
<td>Develop a logical model of the new application system</td>
<td>Define general information system requirements</td>
</tr>
<tr>
<td>Prepare external system design</td>
<td></td>
</tr>
<tr>
<td><strong>Design</strong></td>
<td>Perform preliminary design</td>
</tr>
<tr>
<td>Perform detailed design</td>
<td>Design system test</td>
</tr>
<tr>
<td>Design user aids</td>
<td>Design conversion system</td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td>Set up the development environment</td>
</tr>
<tr>
<td>Code modules</td>
<td>Develop user aids</td>
</tr>
<tr>
<td>Conduct system test</td>
<td></td>
</tr>
<tr>
<td><strong>Facilities Engineering</strong></td>
<td>Gather data</td>
</tr>
<tr>
<td>Conduct site survey</td>
<td>Document facility requirements</td>
</tr>
<tr>
<td>Design data center</td>
<td>Plan site preparation</td>
</tr>
</tbody>
</table>
Prepare site
Plan hardware installation
Install and test hardware

*Implementation*

- Develop contingency procedures
- Develop maintenance and release procedures
- Train system users
- Ensure that production environment is ready
- Convert existing data
- Install application system
- Support acceptance test
- Provide warranty support

*Post-implementation*

- Initiate support and maintenance

*Support*

- Services
  - Support hardware and communication configuration
  - Support software
  - Perform other project completion tasks as appropriate

### 7.3 Project Management

Many factors contribute to a project's success. One of the most essential is establishing an effective project control and reporting system.

Sound project control practices not only increase the likelihood of achieving planned project goals but also promote a working environment where the morale is high and the concentration is intense.

This is particularly critical today when technology is so fluid and the need for isolating the developer from the specific technology is so significant.

The objectives of effective project management are as listed below:

1. *Plan the project:*
   - Define project scope
   - Define deliverables
   - Enforce methodology
Client/Server Architecture

Identify tasks and estimates
Establish project organization and staffing
Document assumptions
Identify client responsibilities
Define acceptance criteria
Define requirements for internal quality assurance review
Determine project schedules and milestones
Document costs and payment terms

2. Manage and control project execution:

Maintain personal commitment
Establish regular status reporting
Monitor project against approved milestones
Follow established decision and change request procedures log and follow up on problems

3. Complete the project:

Establish clear, unambiguous acceptance criteria
Deliver a high-quality product consistent with approved criteria
Obtain clear acceptance of the product

New technologies such as client/server place a heavy burden on the architecture definition phase. The lack of experience in building client/server solutions, combined with the new paradigm experienced by the user community, leads to considerable prototyping of applications. These factors will cause rethinking of the architecture. Such a step is reasonable and appropriate with today's technology.

The tools for prototyping in the client/server platform are powerful enough that prototyping is frequently faster in determining user requirements than traditional modeling techniques were.

When an acceptable prototype is built, this information is reverse engineered into the CASE tool's repository. Bachman's Information Systems' CASE
products provide among the more powerful available tools to facilitate this process.

7.4 Architecture Definition

The purpose of the architecture definition phase in the methodology is to define the application architecture and select the technology platform for the application. To select the application architecture wisely, you must base the choice on an evaluation of the business priorities. Your organization must consider and weight the following criteria:

- Cost of operation—How much can the organizations afford to pay?
- Ease of use—Are all system users well-trained, computer literate, and regular users? Are some occasional users, intimidated by computers, users with little patience, or familiar with another easy to use system? Will the public in situations that don’t allow for training or in which mistakes are potentially dangerous use the system?
- Response time—What is the real speed requirement? Is it less than 3 seconds 100 percent of the time? What is the impact if 6 percent of the time the response lag is up to 7 seconds?
- Availability—What is the real requirement? Is it 24 hours per day, 7 days per week, or something less? What is the impact of outages? How long can they last before the impact changes?
- Security—What is the real security requirement? What is the cost or impact of unauthorized access? Is the facility secure? Where else can this information be obtained?
- Flexibility to change—How frequently might this application change? Does marketing priorities, legislative changes, or technology changes drive the system?
- Use of existing technology—What is the existing investment? What are the growth capabilities? What are the maintenance and support issues?
- System interface—What systems must this application deal with? Are these internal or external? Can the systems being interfaced be modified?
These application architecture issues must be carefully evaluated and weighed from a business perspective. Only after completing this process can managers legitimately review the technical architecture options. They must be able to justify the technology selection in the way it supports the business priorities.

Once managers understand the application architecture issues, it becomes appropriate to evaluate the technical architecture options. Notice that staff are not yet selecting product, only architectural features. It is important to avoid selecting the product before purchasers understand the baseline requirements.

The following is a representative set of technical architecture choices:

- **Hardware (including peripherals)**—Are there predefined standards for the organization? Are there environmental issues, such as temperature, dirt, and service availability?
- **Distributed versus centralized**—Does the organization have a requirement for one type of processing over the other? Are there organizational standards?
- **Network configuration**—Does the organization have an existing network? Is there a network available to all the sites? What is the capacity of the existing network? What is the requirement of the new one?
- **Communications protocols**—What does the organization use today? Are there standards that must be followed?
- **System software**—What is used today? Are there standards in place? What options are available in the locale and on the anticipated hardware and communications platforms?
- **Database software**—Is there a standard in the organization? What exists today?
- **Application development tools (for example, CASE)**—What tools are in use today? What tools are available for the candidate platforms, database engine, operating system, and communications platforms?
• Development environment—Does such an environment exist today? What standards are in place for users and developers? What other platform tools are being considered? What are the architectural priorities related to development?

• Application software (make or buy, package selection, and so on)
  
  Does the organization have a standard?
  
  How consistent is this requirement with industry-standard products?
  
  If there is a product, what platforms does it run on?
  
  Are these consistent with the potential architecture here? How viable is the vendor?
  
  What support is available?
  
  Is source code available?
  
  What are the application architecture requirements related to product acquisition?

• Human interface—
  
  What are the requirements?
  
  What is in place today?
  
  What are users expecting?

One should not drive the other. It is unrealistic to assume that the application architects can ignore the technical platform, but they should understand the business priorities and work to see that these are achieved. Interfaces must isolate the technical platform from the application developers. These interfaces offer the assurance that changes can be made in the platform without affecting functioning at the application layer.

With the technical architecture well defined and the application architecture available for reference, you're prepared to evaluate the product options. The
selection of the technology platform is an important step in building the SDE.

There will be ongoing temptation and pressure to select only the "best products." However, the classification of "best product in the market," as evaluated in the narrow perspective of its features versus those of other products in a category, is irrelevant for a particular organization.

Only by evaluating products in light of the application and technical architecture in concert with all the products to be used together can you select the best product for your organization.

Architectures and platforms should be organizational. There is no reason to be constantly reevaluating platform choices. There is tremendous benefit in developing expertise in a well-chosen platform and getting repetitive benefit from reusing existing development work.

7.4 Systems Development Environment (SDE)

Once your organization has defined its application and technical architectures and selected its tools, the next step is to define how you'll use these tools. Developers do not become effective system builders because they have a good set of tools; they become effective because their development environment defines how to use the tools well.

An SDE comprises hardware, software, interfaces, standards, procedures, and training that are selected and used by an enterprise to optimize its information systems support to strategic planning, management, and operations.

- An architecture definition should be conducted to select a consistent technology platform.

- Interfaces that isolate the user and developer from the specifics of the technical platform should be used to support the creation of a single-system image.

- Standards and standard procedures should be defined and built to provide the applications with a consistent look and feel.
• Reusable components must be built to gain productivity and support a single-system image.

• Training programs must ensure that users and developers understand how to work in the environment.

IBM defined its SDE in terms of an application development cycle, represented by a product line it called AD/Cycle.

The SDE must encompass all phases of the systems development life cycle and must be integrated with the desktop. The desktop provides powerful additional tools for workstation users to become self-sufficient in many aspects of their information-gathering needs.

The most significant advantages are obtained from an SDE when a conscious effort is made to build reusable components. With the uncertainty surrounding product selection for client/server applications today, the benefits of using an SDE to isolate the developers from the technology are even more significant.

These technologies will evolve, and we can build applications that are isolated from many of the changes.

The following components should be included in any SDE established by an organization:

• **Built-in navigation**—Every process uses the same methods to move among processes. For every process a default next process is identified, and all available processes are identified. A business analyst and not the developer do this navigation definition. Every user and every developer then views navigation in the same way.

• **Standardized screen design**—Well-defined standards are in place for all function types, and these screens are generated by default based on the business process being defined. Users and developers become familiar with the types of screens used for help, add, change, delete, view, and table management functions.

• **Integrated help**—A standardized, context-sensitive help facility should respond to the correct problem within the business process. No
programmer development is required. The end-user and analyst who understand how the system user will view the application provide the help text. Help text is user maintainable after the system is in production.

- **Integrated table maintenance**—Tables are a program design concept that calls for common reference data, such as program error codes, printer control codes, and so on, to be stored in a single set of files or databases. A single table maintenance function is provided for all applications in the organization. Programmers and users merely invoke its services. Thus, all applications share standard tables.

- **Comprehensive security**—A single security profile is maintained for each authorized user. Navigation is tied to security; thus, users only see options they are eligible to use. Every programmer and user see the same security facilities. Security profiles are maintained by an authorized user and use the table maintenance facilities.

- **Automatic view maintenance**—Screens are generated, navigation is prescribed, and skeleton programs are generated based on the security profile and business requirements defined for a process. The developer does not have to write special code to extract data from the database. All access is generated based on the defined business processes and security.

- **Standard skeleton programs**—An analyst answers a set of questions to generate a skeleton program for each business process. This feature includes standard functions that the programmer will require.

Every platform includes a set of services that are provided by the tools. This is particularly true in the client/server model, because many of the tools are new and take advantage of object-oriented development concepts. It is essential for an effective SDE to use the facilities and not to redevelop these.

The advantages of building an SDE and including these types of components are most evident in the following areas:
Client/Server Architecture

- **Rapid prototyping**—The development environment generates skeleton applications with embedded logic for navigation, database views, security, menus, help, table maintenance, and standard screen builds. This framework enables the analyst or developer to sit with a user and work up a *prototype of the application rapidly*. In order to get business users to participate actively in the specification process, it is necessary to show them something real.

- A *prototype* is more effective for validating the process model than are traditional business modeling techniques. Only through the use of an SDE is such prototyping possible. Workstation technology facilitates this prototyping. The powerful GUI technology and the low cost of direct development at the workstation make this the most productive choice for developing client/server applications.

- **Rapid coding**—Incorporating standard, reusable components into every program reduces the number of lines of custom code that must be written. In addition, there is a substantial reduction in design time, because much of the design employs reusable, standard services from the SDE. The prototype becomes the design tool.

- **Consistent application design**—As mentioned earlier, much of the design is inherent in the SDE. Thus, by virtue of the prototype, systems have a common look and feel from the user's and the developer's perspectives. This is an essential component of the single-system image.

- **Simplified maintenance**—The standard code included with every application ensures that when maintenance is being done the program will look familiar. Because more than 60 percent of most programs will be generated from reusable code, the maintenance programmer will know the modules and will be able to ignore them unless global changes are to be made in these service functions. The complexity of maintenance corresponds to the size of the code and the amount of familiarity the programmer has with the program source. The use of reusable code provides the programmer with a common look and much less new code to learn.

- **Enhanced performance**—Because the reusable components are written once and incorporated in many applications, it is easier to justify getting the best developers to build the pieces. The ability to
make global changes in reusable components means that when performance problems do arise, they can often be fixed globally with a single change.

7.6 CASE

CASE tools are built on an "enterprise model" of the processes to be automated; that is, systems integration and software development. This underlying enterprise model or "metamodel" used by CASE is crucial to the tool's usefulness. Tools based on a poor model suffer from poor integration, are unable to handle specific types of information, require duplicate data entry, cannot support multiple analyst-developer teams, and are not flexible enough to handle evolving new techniques for specifying and building systems solutions. Tools with inadequate models limit their users' development capabilities.

All the leading CASE products operate and are used in a client/server environment. Intel 486-based workstations operating at 60MHz or faster, with 16-24 Mbytes of memory and 260Mbyte hard disks and UNIX workstations of similar size are typically required. Thus, combining hardware and CASE software costs, CASE costs up to $20,000 per user workstation/terminal.

Unfortunately, a thorough review of the available CASE products shows that none adequately provide explicit support for development of client/server applications and GUIs. This lack of support occurs despite the fact that they may operate as network-based applications that support development of host-based applications. There is considerable momentum to develop products that support the client/server model. The Bachman tools are in the forefront in this area because of their focus on support for business process reengineering. With many client/server applications being ported from a minicomputer or mainframe, the abilities to reuse the existing models and to reverse engineer the databases are extremely powerful and timesaving features.

It seems likely that no single vendor will develop the best-integrated tool for the entire system's life cycle. Instead, in the probable scenario, developers mix the best products from several vendors. IBM envisions this scenario in their AD/Cycle product line, by Computer Associates in their CA90 products, and by NCR in their Open Cooperative Computing series of products.
As an example, an organization may select Bachman, which provides the best reengineering, and reusability components and the only true enterprise model for building systems solutions for their needs. This model works effectively in the LAN environment and supports object-oriented reuse of specifications. The organization then integrates the Bachman tools with ParcPlace Parts product for Smalltalk code generation for Windows, UNIX or OS/2 desktops and server applications and with Oracle for code generation in the UNIX, OS/2, and Windows NT target environment. The visual development environments of these products provide the screen painting, business logic relationship, and prototyping facilities necessary for effective systems development.

A more revolutionary development is occurring as CASE tools like the Bachman products are being integrated with development tools from other vendors. These development tools, used with an SDE, allow applications to be prototyped and then reengineered back into the CASE tool to create process and data models. With the power of GUI-based development environments to create and demonstrate application look and feel, the prototyping approach to rapid application design (RAD) is the only cost-effective way to build client/server applications today.

Users familiar with the ease of application development on the workstation will not accept paper or visual models of their application. They can only fully visualize the solution model when they can touch and feel it. This is the advantage of prototyping, which provides a "real touch and feel." Except in the earliest stages of solution conceptualization, the tools for prototyping must be created using the same products that are to be used for production development.

Not all products that fall into the CASE category are equally effective. For example, some experts claim that the information engineering products—such as Texas Instruments' product, IEF—attempt to be all things to all people. The criticism is that such products are constrained by their need to generate code efficiently from their models. As a result, they are inflexible in their approach to systems development, have primitive underlying enterprise models, may require a mainframe repository, perform poorly in a team environment, and provide a physical approach to analysis that is constrained by the supported target technologies (CICS/DB2 and, to a lesser extent, Oracle). Critics argue that prototyping with this class of tool requires
developers to model an unreasonable amount of detail before they can present the prototype.

### 7.7 Object-Oriented Programming (OOP) and CASE Tools

OOP is a disciplined programming style that incorporates three key characteristics: encapsulation, inheritance, and dynamic binding. These characteristics differentiate OOP from traditional structured programming models, in which data has a type and a structure, is distinct from the program code, and is processed sequentially. OOP builds on the concepts of reuse through the development and maintenance of class libraries of objects available for use in building and maintaining applications.

- **Encapsulation** joins procedures and data to create an object, so that only the procedures are visible to the user; data is hidden from view. The purpose of encapsulation is to mask the complexity of the data and the internal workings of the object. Only the procedures (methods) are visible to the outside world for use.

- **Inheritance** passes attributes to dependent objects, called descendants, or receives attributes from objects, called ancestors, on which the objects depend. For example, the family airplane includes all structures, whereas the descendant jet inherits all the properties of airplane and adds its own, such as being no propeller-driven; the child F14 inherits all the properties of airplane and jet and adds its own properties—speed greater than 1,400 mph and climbing rate greater than 60 feet per second.

- **Dynamic binding** is the process whereby linking occurs at program execution time. All objects are defined at runtime, and their functions depend on the application's environment (state) at the time of program execution. For example, in a stock management application, the function called program trading can sell or buy, depending on a large range of economic variables that define the current state. These variables are transparent to the user who invokes the trade process.

- **Class library** is a mature, tested library of reusable code that provides application-enabling code such as help management, error recovery, function key support, navigation logic, and cursor management. The class library concept is inherent to the SDE concept and—in combination with the standards and training fundamental—is inherent
to the productivity and error reductions encountered in projects that use an SDE.

Object-oriented programming is most effective when the reusable components can be cut and pasted to create a skeleton application. Into this skeleton the custom business logic for this function is embedded. It is essential that the standard components use dynamic binding so that changes can be made and applied to all applications in the environment. This provides one of the major maintenance productivity advantages.

Certain programming languages are defined to be object-oriented. C++, Objective C, SmallTalk, MacApp, and Actor are examples. With proper discipline within an SDE it is possible to gain many of the advantages of these languages within the more familiar environments of COBOL and C. Because the state of development experience in the client/server world is immature, it's imperative for organizations to adopt the discipline of OOP to facilitate the reuse of common functions and to take advantage of the flexibility of global changes to common functions.

Objects are easily reused, in part because the interface to them is so plainly defined and in part because of the concept of inheritance. A new object can inherit characteristics of an existing object "type." You don't have to reinvent the wheel; you can just inherit the concept. Inheritance gives a concise and precise description of the world and helps code reusability, because every program is at the level in the "type hierarchy" at which the largest number of objects can share it. The resulting code is easier to maintain, extend, and reuse.

A significant new component of object-oriented development has been added with the capability to use server objects with RPC requests.

During 1994, the introduction of CORBA compliant object stores will dramatically open the client/server paradigm to the "anything anywhere" dimensions. Objects will be built and stored on an arbitrary server for use by any client or server anywhere.

The earliest implementations of this model are provided by NeXT with its Portable Distributed Objects (PDO) and Suns Distributed Objects Everywhere (DOE) architecture.

And what about object-oriented database management system (OODBMS)?
It combines the major object-oriented programming concepts of data abstraction, encapsulation, and type hierarchies with the database concepts of storage management, sharing, reliability, consistency, and associative retrieval.

When is an OODBMS needed, and when will an extended relational database management system (DBMS) do?

Conventional database management products perform very well for many kinds of applications. They excel at processing large amounts of homogeneous data, such as monthly credit card billings.

They are good for high-transaction-rate applications, such as ATM networks. Relational database systems provide good support for ad hoc queries in which the user declares what to retrieve from the database as opposed to how to retrieve it.

As we traverse the 1990s, however, database management systems are being called on to provide a higher level of database management. No longer will databases manage data; they must manage information and be the knowledge centers of the enterprise.

To accomplish this, the database must be extended to

- Provide a higher level of information integration
- Store and retrieve all types of data: drawings, documents, fax, images, pictures, medical information, voice, and video

Many RDBMS products already handle binary large objects (BLOBs) in a single field of a relation. Many applications use this capability to store and provide SQL-based retrieval of digital laboratory data, images, text, and compound documents.

*Digital's Application Driven Database Systems (ADDS)* have been established to enable its SQL to handle these complex and abstract data types more explicitly and efficiently.

But applications that require database system support are quickly extending beyond such traditional data processing into computer-aided design (CAD) and CASE, sophisticated office automation, and artificial intelligence.
These applications have complex data structuring needs, significantly different data accessing patterns, and special performance requirements. Conventional programming methodologies are not necessarily appropriate for these applications and conventional data management systems may not be appropriate for managing their data.

Consider for a moment the factors involved in processing data in applications such as CAD, CASE, or generally in advanced office automation. The design data in a mechanical or electrical CAD database is heterogeneous. It consists of complex relationships among many types of data.

The transactions in a CASE system don't lend themselves to transaction-per-second measurement; transactions can take hours or even days. Office automation applications deal with a hierarchical structure of paragraphs, sentences, words, characters, and character attributes along with page position and graphical images.

Database access for these applications is typically a directed graph structure rather than the kind of ad hoc query that can be supported in SQL. Each object contains within its description reference to many other objects and elements. The object manager to provide the total view automatically collects these. In typical SQL applications, the developer makes explicit requests for related information.

In trying to manipulate such complex data using a relational system, a programmer writes code to map extremely complex in-memory data structures onto lower-level relational structures using awkward and resource-intensive recursive programming techniques.

The programmer finds himself or herself doing database management instead of letting the DBMS handle it. Worse, even if the programmer manages to code the translation from in-memory objects to relational tables, performance is unacceptable.

Thus, relational systems have not been any help for the programmer faced with these complex coding tasks. The object-oriented programming paradigm, on the other hand, has proven extremely useful. The complex data structures CAD and CASE programmers deal with in memory are often defined in terms of C++ or Smalltalk objects.
It would be helpful if the programmer didn't have to worry about managing these objects, moving them from memory to disk, then back again when they're needed later. Some OOP systems provide this object "persistence" just by storing the memory image of objects to disk. But that solution only works for single-user applications. It doesn't deal with the important concerns of multiuser access, integrity, and associative recall.

*Persistence* means that objects remain available from session to session. Reliable means automatic recovery in case of hardware or software failures. Sharable means that several users should be able to access the data.

All of these qualities may require systems that are larger than many that are currently available. In some cases, of course, programmers aren't dealing with overwhelmingly complex data, yet want to combine the increased productivity of object-oriented programming with the flexibility of an SQL DBMS.

Relational technology has been extended to support binary large objects (BLOBs), text, image and compound documents, sound, video, graphics, animation, and abstract data types.

As a result, organizations will be able to streamline paper-intensive operations to increase productivity and decrease business costs—assuming they use a database as a repository and manager for this data.

---

**UNIT 8**

**CORBA**

**8.1 Common Request Broker Architecture (CORBA)**

8.1.1 OMG (Object Management Group)

8.1.2 CORBA .......... what is it?
8.1 Common Request Broker Architecture

8.1.1 OMG (Object Management Group)

OMG was created in 1989. The OMG solicited input from all segments of the industry and eventually defined the CORBA standards. OMG has been an international, open membership, not-for-profit computer industry consortium since 1989. Any organization may join OMG and participate in standards-setting process.

OMG’s modeling standards enable powerful visual design, execution and maintenance of software and other processes. OMG’s middleware standards and profiles are based on the Common Object Request Broker Architecture (CORBA®) and support a wide variety of industries.
8.1.2 CORBA ……. what is it?

Distributed Object Systems

*Distributed object systems* are distributed systems in which all entities are modeled as objects. Distributed object systems are a popular paradigm for object-oriented distributed applications. Since the application is modeled as a set of cooperating objects, it maps very naturally to the services of the distributed system.

*CORBA, or Common Object Request Broker Architecture, is a standard architecture for distributed object systems. It allows a distributed, heterogeneous collection of objects to interoperate.*

OR

*CORBA is the acronym for Common Object Request Broker Architecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks.*

- It uses the standard protocol IIOP. A CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.
- The CORBA specification is supported by more than 700 hardware and software manufacturers, government organizations and user groups, creating a rich and robust framework that successfully operates across heterogeneous computing platforms.
- CORBA is proving to be one of the most important innovations in the recent history of information systems. It addresses the two most prominent problems faced in the software industry today:
  - The difficulty of developing client server applications.
  - By rapidly integrating legacy systems, off-the-shelf applications and the new development.
- CORBA is the specification for an emerging technology known as distributed object management (DOM), where DOM provides a higher-level, object-oriented interface on top of the basic distributed computing services.
• Microsoft’s Distributed Component Object Model (DCOM) is the only competitor for CORBA among the DOM solutions.
• At the basic level, CORBA defines a standard framework from which an information system implementer of software developer can easily and quickly integrate network-software modules and applications to create new and more powerful applications.
• The term CORBA is often used to refer to ORB (Object Request Broker) itself, as well as to the entire OMG architecture.

8.1.3 Who is using CORBA already?

Thousands of sites rely on CORBA for enterprise, Internet, and other computing. The Object Management Group maintains an entire website devoted to user design wins and success stories.

8.1.4 What is CORBA good for?

CORBA is useful in many situations. Because of the easy way that CORBA integrates machines from so many vendors, with sizes ranging from mainframes through minis and desktops to hand-held’s and embedded systems, it is the middleware of choice for large (and even not-so-large) enterprises.

One of its most important, as well most frequent, uses is in servers that must handle large number of clients, at high hit rates, with high reliability.

CORBA works behind the scenes in the computer rooms of many of the world's largest websites; ones that you probably use every day.

Specializations for scalability and fault-tolerance support these systems. But it's not used just for large applications; specialized versions of CORBA run real-time systems, and small-embedded systems.

8.1.6 ORB and CORBA Architecture

The ORB

• The ORB is the distributed service that implements the request to the remote object. It locates the remote object on the network, communicates the request to the object, waits for the results and when available communicates those results back to the client.
• The ORB implements location transparency. The client and the CORBA object regardless of where the object is located use exactly the same request mechanism. It might be in the same process with the client, down the hall or across the planet. The client cannot tell the difference.
• The ORB implements programming language independence for the request. The client issuing the request can be written in a different programming language from the implementation of the CORBA object.
• The ORB does the necessary translation between programming languages. Language bindings are defined for all popular programming languages.

**CORBA Architecture**

• CORBA defines architecture for distributed objects. The basic CORBA paradigm is that of a request for services of a distributed object. Everything else defined by the OMG is in terms of this basic paradigm.
• The services that an object provides are given by its *interface*. Interfaces are defined in OMG's Interface Definition Language (IDL). Distributed objects are identified by object references, which are typed by IDL interfaces.
• The figure 8.1.6 below graphically depicts a request. A client holds an object reference to a distributed object. The object reference is typed by an interface. In the figure 8.1.6 below the object reference is typed by the Rabbit interface.
• The Object Request Broker, or ORB, delivers the request to the object and returns any results to the client.
• In the figure, a jump request returns an object reference typed by the AnotherObject interface.
OMG Reference Model Architecture

- **Object Services** -- These are domain-independent interfaces that are used by many distributed object programs.
  - For example, a service providing for the discovery of other available services is almost always necessary regardless of the application domain. Two examples of Object Services that fulfill this role are:
- The *Naming Service* -- which allows clients to find objects based on names;
- The *Trading Service* -- which allows clients to find objects based on their properties.

There is also Object Service specifications for lifecycle management, security, transactions, and event notification, as well as many others

- **Common Facilities** -- Like Object Service interfaces, these interfaces are also horizontally oriented, but unlike Object Services they are oriented towards end-user applications.
  - An example of such a facility is the *Distributed Document Component Facility* (DDCF), a compound document Common Facility based on OpenDoc. DDCF allows for the presentation and interchange of objects based on a document model, for example, facilitating the linking of a spreadsheet object into a report document.

- **Domain Interfaces** -- These interfaces fill roles similar to Object Services and Common Facilities but are oriented towards specific application domains.
  - For example, one of the first OMG RFPs issued for Domain Interfaces is for Product Data Management (PDM) Enablers for the manufacturing domain.

- **Application Interfaces** - These are interfaces developed specifically for a given application. Because they are application-specific, and because the OMG does not develop applications (only specifications), these interfaces are not standardized. However, if over time it appears that certain broadly useful services emerge out of a particular application domain, they might become candidates for future OMG standardization.
8.1.6 Technical Details of CORBA

CORBA applications are composed of objects, individual units of running software that combine functionality and data, and that represent something in the real world.

Typically, there are many instances of an object of a single type - for example, an e-commerce website would have many shopping cart object instances, all identical in functionality but differing in that each is assigned to a different customer, and contains data representing the merchandise that its particular customer has selected.

For other types, there may be only one instance. When a legacy application, such as an accounting system, is wrapped in code with CORBA interfaces and opened up to clients on the network, there is usually only one instance.

For each object type, such as the shopping cart that we just mentioned, you define an interface in OMG IDL.

The interface is the syntax part of the contract that the server object offers to the clients that invoke it. Any client that wants to invoke an operation on the object must use this IDL interface to specify the operation it wants to perform, and to marshal the arguments that it sends.

When the invocation reaches the target object, the same interface definition is used there to unmarshal the arguments so that the object can perform the requested operation with them. The interface definition is then used to marshal the results for their trip back, and to unmarshal them when they reach their destination.

The IDL interface definition is independent of programming language, but maps to all of the popular programming languages via OMG standards: OMG has standardized mappings from IDL to C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript.

This separation of interface from implementation, enabled by OMG IDL, is the essence of CORBA. The interface to each object is defined very strictly.

In contrast, the implementation of an object - its running code, and its data - is hidden from the rest of the system (that is, encapsulated) behind a boundary that the client may not cross.
Clients’ access objects only through their advertised interface, invoking only those operations that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation.

8.2 CORBA Object Services

The CORBA Services are collection of system-level services packaged with IDL-specifed interfaces. These object services augments and complements the functionality of ORB.

The OMG has published standards for fifteen object services listed below:

1. **The Life Cycle Service** - enlists the operations for creating, copying, moving and deleting components on the bus.

2. **The Persistence Service** – provides a single interface for storing components persistently on a variety of storage servers – including Object Databases (ODBMS’s), Relational Databases (RDBMS’s) and simple files.

3. **The Naming Services** – allows components on the bus locate other components by name; it also supports federated naming contexts. This service also allows objects to be bound to existing network directories or naming contexts – including ISO’s X.600, OSF’a DEC, Sun’s NIS+, Novell’s NDS and the Internet’s LDAP.

4. **The Event Service** - allows components on the bus to dynamically register or unregister their interest in specific events. It defines a well-known object called an event channel that collects and distributes events among components that knows nothing about each other.

5. **The Concurrency Control Service** - provides a lock manager that can obtain locks on behalf of the transactions or threads.
6. **The Transaction Service** – provides two-phase commit coordination among recoverable components using either flat or nested transactions.

7. **The Relationship Service** – provides a way to create dynamic association between components that know nothing of each other. It also provides mechanisms for traversing the links that group these components. This service can be used to enforce referential integrity constraints, track containment relationships and for any type of linkage among components.

8. **The Externalization Service** – provides a standard way of getting data into and out of a component using a stream-like mechanism.

9. **The Query Service** – it provides query operations for objects. It is a superset of SQL. It is based on the SQL3 specification and the Object Database Management Group’s (ODMG) Object Query Language (OQL).

10. **The Licensing Service** – provides operations for metering the use of components to ensure fair compensation for their use, the service supports any model of usage control any point in a component’s life cycle. It supports charging per session, per node, per instance creation and per site.

11. **The Properties Service** – provides operations that let you associate named values or properties with any component. Using this service, you can dynamically associate properties with a component’s state like a title or a date.

12. **The Time Service** – provides interfaces for synchronizing time in a distributed object environment. It also provides operations for defining and managing time-triggered events.
13. **The Security Service** – provides a complete framework for distributed object security. It supports authentication, access control lists, confidentiality and non-repudiation. It also manages the delegation of credentials between objects.

14. **The Trader Service** - provides a “Yellow Pages“ for the objects; it allows objects to publicize their services and bid for jobs.

15. **The Collection Service**- provides CORBA interfaces to generically create and manipulate the most common collections.

The fig 8.2 shows the Request For Proposal (RFP) schedules that OMG is using to develop the object service specifications. OMG RFPs are requests for a technology.

Usually an RFP is met by emerging the responses obtained form several organizations. The working standard can be obtained in 12 to 16 months.

Now OMG has completed its work on object services and ORB specifications, releasing CORBA 3.0, now the focus is on CORBA Beans, CORBA domains, and Business Objects.
8.3 CORBA Common Facilities

The CORBA facilities are collections of IDL defined frameworks that provide services of direct use to application objects.

OR

It is a collection of services that many applications may share, but which are not as fundamental as the Object Services. For instance, a system management or electronic mail facility could be classified as a common facility.

Common Facilities are divided into two major categories:

1. Horizontal Common Facilities - which are used by most systems,
2. Vertical Market Facilities - which are domain-specific.

**Horizontal Common Facilities**

Includes the user interface, information management, system management, and task management.

The Horizontal Common Facilities are those facilities that are used by most systems.

**Vertical Market Facilities**

Are those facilities that are specific to particular domains or industries, rather than widely applicable.

**8.4 CORBA 3.0: Next Generation**

Formally, CORBA 2 and CORBA 3 refer to complete releases of the entire CORBA specification. However, because OMG increments the major release number only when they make a significant addition to the architecture, these phrases become a sort of shorthand for just the significant addition.

*CORBA seem to be under construction, but it is moving at jet speed to keep up with the requirements of the Object Web.*

*By Sep 1998, OMG announced CORBA 3.0. CORBA 3.0 is much more integrated with the Java Object Model. It now has IDL-to-Java mapping as well as a reverse Java-to-IDL mapping which allows you to start with Java RMI semantics and produce CORBA stubs and skeletons.*

*CORBA is also becoming more like Java with its support for the Multiple Interfaces and Object-by-Value.*

CORBA has also adopted the EJB component model with extensions for multiple-language support.

CORBA is also being augmented with a Common Facility for Mobile Agents, a Scripting Engine specification and a Workflow Facility. The following figure 8.4 depicts the evolution of CORBA.
So, "CORBA 2" sometimes refers to CORBA interoperability and the IIOP protocol, and "CORBA 3" sometimes refers to the CORBA Component Model CCM (even though the term CORBA 3 really refers to a suite of ten specifications!).

**CORBA 2.0**
*Intergalactic ORB’s*
- IIOP
- Federated IR
- C++ bindings
- Transactions
- Concurrency
- Externalization
- Relationships
- Query
- Licensing
- Compound Documents
- Trader
- Security
- Collections

**CORBA 3.0**
*Server Components*
- Messaging (MOM)
- Server Portability (POA)
- Multiple Interfaces
- CORBA Beans/Business Objects
- Java Bindings
- RMI-over-IIOP
- Objects-by-Value
- Mobile Agents
- CORBA/DCOM
- Automatic Persistence
- IIOP Firewall Support
- Workflow
- Domain-level Framework

---

**Fig 8.4 the Evolution of CORBA**

**8.6 DCOM Objects (Distributed Component Object Model)**

**8.6.1 Component Object Model (COM) History**

In 1990, Microsoft introduced the OLE 1 technology as its basic strategy for integrating multiple applications and multimedia data types within a compound document framework.
OLE 1 was a clumsy and slow protocol built on top of DDE. When an application was launched from within a compound document, it was given its own window, and the data it needed was copied into its address space. There was no way for the application to directly access the document.

Then OLE 2 introduced in 1993, fixed many of these shortcomings with a new object-encapsulation technology called the Component Object Model (COM). All of OLE2 is built on top of COM.

Here Microsoft added a new custom, control architecture OCX to the compound document framework.

Whereas in early 1996, Microsoft announced ActiveXs, which is minimalist OLE objects for the Internet. During the same time, Microsoft released the first distributed version of COM called as the Distributed Component Object Model (DCOM).

DCOM was part of the NT4.0, and then in 1997 Microsoft announced COM+ which formed the next generation of COM. With this it also introduced a new marketing architecture called the Distributed internet Application Architecture (DNA).

8.6.2 About COM

COM separates the object interface from its implementation and requires that all interfaces be declared using an Interface Definition Language (IDL).

The Microsoft’s IDL is based on DCE it is of course, not CORBA compliant.

The COM object is not an object in the Object Oriented sense. COM objects do not have a persistent identity, where a COM object reference is simply an interface pointer to an object in memory.

i.e., COM clients are given a pointer to access the functions in an interface; this pointer is not related to state information. A COM client cannot reconnect to exactly the same object instance with the same state at a later time.

COM provides both the static and dynamic interfaces for method invocation.
Com interface is a collection of function calls also known as methods or member functions. The COM interface serves as a client/server contract. It defines the functions independently from their implementation.

COM interfaces are language independent and you can use it to call functions across address spaces and networks. Microsoft provides COM language bindings for all its development environments including Visual C++, Visual Basic and Visual J++.

A COM interface is defined as a low-level binary API based on a table of pointers. To access an interface, COM clients use pointers to an array of function pointers which are also known as virtual table or vtable.

Each COM object has one or more vtables that define the contract between the object implementation and its clients.

**8.6.3 What is DCOM?**

This is a different flavor of COM. DCOM allows your component to run on a computer other than the one that is using your DCOM component; whereas a standard COM component runs on the same machine and in the same process space as the software that uses it.

To master DCOM is to understand the details of how they affect the software you are building.

**8.6.4 Why use DCOM?**

It is your ticket to the distributed part of your application; it provides you with ability to talk to services on a machine other than the client computer. In the past, all the functionality was on the host machine and perhaps the database on the server.

But with DCOM, you can now place actual functionality on the server and access it almost as if they were local.

**8.6.5 Benefits of DCOM**
Client/Server Architecture

a. Write components once and allow many programmers to use it.
b. You can design and implement only the components’ interface and other programmers can code to that interface.
c. When component’s changes, you can replace them once on a server instead of on every client machine.
d. All clients get the update when the server-side component is in place.

8.6.6 What is not good in DCOM?

- As a programmer, you need to change and focus from being property-oriented to service-oriented. For this you need thinking in different perspective that takes some time.

- Installation of DCOM components is more involved than traditional application installation. You must be sure that the client knows about the client-side components and where they are located.

- Components located on a network can be slower than components located on a server due to the network traffic turn-around time.

8.6.7 Basic Concepts of DCOM

The DCOM has some basic concept associated with it. They are

1. **DCOM Component Locations**

   The COM components are created as an ActiveX DLL, which has the Classes your program can access through the COM components.

   The components are registered on the client system in the Windows Registry. The Registry entries also give the picture on where exactly the DLL file is located on the machine, so that it is loaded and executed.

   The DCOM components still have to be registered as they reside on different machine. This tells the information about the real location on the remote computer, the component is running, thus now the client knows where the
component is. When a component is called, from the programmer’s view an instance of the DCOM component is created.

The runtime system looks for the Registry entry of the component which we are trying to create, if found it connects to the machine across the network and makes the request.

2. In-Process and Out-of-Process Components

*In-process* - components are running in the same process space as its applications, they are built as DLL’s.

Out-of-process – components are running in a different address space from the application. They do not share the same process space, and they are built as EXE’s.

3. The Surrogate

There is an exception point between ActiveX EXE’s and DLL’s. We know that EXE’s are out-of-process components and DLL’s are in-process ones. But when writing components for Microsoft Transaction Server (MTS) or COM+, you need to create DLL’s. This makes it difficult to create a multipurpose components that can run in MTS or as stand-alone server-side, out-of-process components.

In order to address this problem Microsoft created a concept called *Surrogate Process*. This process wraps a DLL component in an EXE so that it can run as an out-of-process DCOM component. This exception will allow you to use components that were built as ActiveX DLL’s on a server as DCOM components.

4. Remote Procedure Call (RPC)

DCOM is entirely dependent on Microsoft RPC to provide the distributed capabilities. *RPC only turns COM into DCOM*. Microsoft infact extends RPC to accommodate DCOM via a mechanism known as the Object RPC (ORPC).
ORPC extended RPC is how the calls are made on remote objects, as well as how references are transmitted and referenced.

It has 3 concepts in it as listed below:

- **RPC as Middleware**
- **RPC Functions**
- **RPC Steps**

### 5. Marshaling

It is an important concept when dealing with DCOM. We know that when you pass data from your application to a regular in-process COM component, the component’s stack is used to store the parameters; similar to the function call in the application.

But the same is not possible for the out-of-process DCOM components; here we need different machines to move the data across the process boundary. This mechanism is called as Marshaling.

The Marshaling works through a mechanism that user objects known as proxies and stubs.

The proxy function is built into the client and a stub function is built into the component. The proxy portion marshals the data and prepares it for shipment across the process boundary to the component.

The stub function in the component unmarshals the data for use by the component.
8.6 COM Server

A COM Server is a piece of code a DLL or an EXE or a Java Class that houses one or more object classes each with its own CLSID (Class ID).

When a client asks for an object of a given CLSID, DCOM loads the server code and asks it to create an object of that class.

The server must provide a class factory for creating a new object. Once a object is created, a pointer to its primary interfaces is returned to the client. The server is not the object itself. The word “server” in DCOM is used to emphasize the serving agent.

A COM server must satisfy the following:

1. **Implement a class factory interface**

   The server must implement a class factory with the IClassFactory interface for each supported CLSID. The class factory creates instance of a class. If a class supports licensing then it must implement the IClassFactory2 Interface.

   This interface creates an object only if a valid license file is present or a license key is provided.

2. **Register the classes it supports**

   The server must register a CLSID for each class it supports with the NT Registry. For each CLSID, it must create one or more entries that provide the pathname to the server DLL or EXE (or to both). The information is recorded using the NT Registry API’s. Generally classes are registered at the installation time.

3. **Initialize the COM library**

   The server issues a call to the COM API CoInitialize to initialize DCOM. The COM library provides run-time services and API’s. These are functions with the Co prefix, which stands for Component.
4. **Verify that the library is of a compatible version**

   The server does this by calling the API CoBuildVersion.

5. **Provide an unloading mechanism**

   The server must provide a way to terminate itself when there are no active clients for its objects.

6. **Uninitialize the COM Library**

   The server calls the COM API CoUninitialize when it is no longer in use.

The implementation of the server – including registration, the class factory and the unloading mechanism will differ depending on whether the serve is packaged as a DLL or EXE or. class file.
Appendix

1. Network Operating System

NOS middleware must provide to create a “single-system image” of all the services on the network. That is the NOS middleware provides the glue that recreate the single system out of the disparate elements. One of the main functions of NOS is to make the physical location of resources over the network transparent to an application.

Early NOS

The early NOS’s were in the business of virtualizing the file and printer resources and redirecting them to LAN-based file and the print servers.

These NOS’s provided proxies on the local machines – the requesters- that intercepted calls for devices and redirected them to servers on the LAN. The only way an application or the user could tell the difference between the local or remote resource was form a pathname, which included its machine name.

The new generation of network file servers promises to introduce even more transparency into the file systems. Like the DCE Distributed File Service (DFS), which provides a single-image file system that can be distributed across a group of file servers.

The DFS provides a Local File System (LFS) with many advanced features, including replication facilities that make the file system highly available. It gives a faster response with the help of the distributed cache. A snapshot of the file system can reside on the client, which can operate on files even if the server is down.

DFS cam work with the other local file systems such as NFS, or the unix file system.

IBM provides DFS servers for multiple platforms including MVS, several Unixes and NT.
Where is NOS heading to?

1. **NOS function are being subsumed by server OSs**
   - The early NOSs like the NetWare 3.X and LAN Manager were mainly in the business of providing shared file and printer access to DOS client machines.
   - The newer OSs like the NT 5.0, NetWare 5.0 and various Unixes are now bundling the NOS functions in the OS.

2. **NOSs are becoming intergalactic**
   - NOSs are transcending their LAN origin and are expanding to reach the global trend including the enterprises, intranet, extranets and the Internet.

3. **Global directories are becoming strategic**
   - In the age of the Intergalactic networking, companies without a global directory will be at a disadvantage.
   - The competing global directories include Novell’s NDS, Banyan’s Universal Street Talk, DCE, Netscape’s Directory Server, and NT 5.0’s forthcoming Active Directory.
   - The next big challenge is to get these directories to interoperate which is where X.500 and LDAP will play a big role.

4. **NOSs are becoming turn-key commodities.**
   - NOS are being delivered as shrink-wrap software in the form of the commodity server OSs.
   - They have become easier to install, use and maintain.
5. **NOSs are becoming multiplatform and Internet savvy**

- We understand that NOS is just an extension of the local OS.
- In the age of the Internet, the NOS itself must be multiplatform, this trend was actually set by the DEC standard.
- Novell ported its best-of-breed NDS directory to other OS platforms including the UNIX and NT.
- Netscape went a step further and introduced a whole suite of multiplatform Internet NOS offerings.
- Finally Java provided a multiplatform OS on top of the OS.
- Thus NOS is also becoming a vehicle for packaging turn key Internet services and related middleware.

Thus NOS is existing from the early days of the client/server computing and is continuously reinventing itself.