3).Three/Multi/n-tier Architecture

1).To create a distributed application that avoid the problems of N/W bottle-necks at the data base server, we should consider using n-tier architecture.

2).This type of application split the solution across several computer.

3).The user interface still reside on the user's computer, and database remain on the data sever.

However, the business rule or data services objects are placed into a component or separate application that reside on the server. Such an application server the two computer perform.

4).In three-tier architecture application is divided into 3(three) elements.

User-Services, Data Services, Business Services where the user-service layer handles all the userinterface issue and handling of data import.

The business services layer manages the handling of business rule or Business logic applied and service layer including, storing and retrieving the data from the database.

2.8 Need for COM in Middle-Tier

In three-tier architecture business rule are generally applied on Middle-tier.

All the business rule cannot be grouped together into a single executable file. Since, the most big organization this strategy will not be suitable as even for a single component all the business rule have to be updated and re-compiled.

As distributed three-tier in architecture in nature, given popularity, COM emerge as the favoured technology.

Due to COM it was possible for developer to create separate component for separate services. These component are independent to each other.

In a three-tier architecture COM component are applied in Middle-tier business rules they are used to implement business services.

These component on independent of each other and have distinct functionality.

3.1 Messaging

1).Messaging is the process of sending completely encapsulated set of data between two application component.

2).Messaging is the loose communication channel between two component.

3).Message can transfer in one or both direction.

4).Message component can exist in same process, in the different process on the same computer, in the different process on the different computer with completely encapsulated set of data.