

Architecture and Application Infrastructure A useful application architecture has been developed by the Gartner Group. It is called the Gartner Three layer model and it breaks the logic of an application down into three layers (or Tiers). They are:

Layer Name	Purpose
Presentation	This layer is responsible for data presentation and user input.
Application	This layer is responsible for all processing performed by the application
Data Management	This layer is responsible for data management and persistence

1 and 2 tier models

Many application structures can be represented within the three layer model. The traditional monolithic application is shown in figure 1. In this case, all three layers are deployed within one application unit (ie. process) on a single machine. Contrast this with the Client-Server model (figure 2), where the the layers are deployed on two separate machines (the client and the server machine).

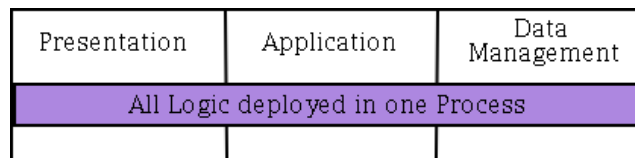


Figure 1: Monolithic Application

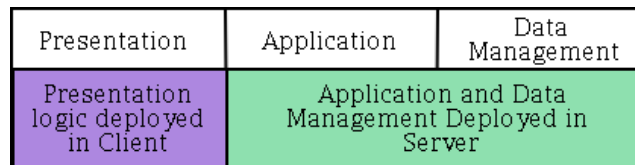


Figure 2: Client/Server Application

In the case of the client server (or two tier use of the model) logic from the layers can be deployed in any configuration between the client and server machine. The terms **thin** client and **fat** client are used to indicate how much of the applications logic has been deployed on the client. If the client is only responsible for managing presentation, then it is typically called a **thin** client. If presentation and application logic is deployed on the client, it is typically called a **fat** client. Figure 3 illustrates the various configurations and the Gartner Label associated with each.

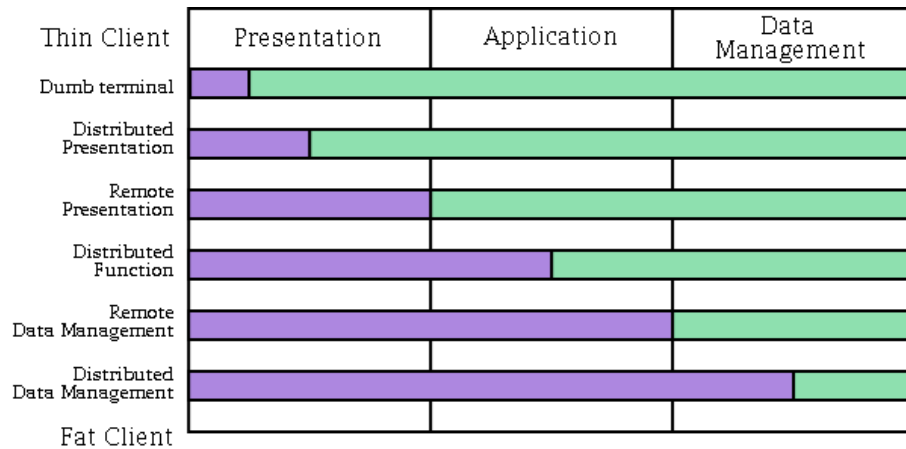


Figure 3: Client/Server configurations

In a client/server model, the server generally represents the Database server. In the case of the Distributed Function configuration, Application logic is deployed on the database server through the use of stored procedures. As more and more of the application logic is coded as stored procedures, the client becomes more and more a thin client. SQL (and its various non-standard forms) has proven to be an excellent data management language, but has not excelled as a language for implementing application logic. SQL's inability to handle complexity well has led to long-term maintenance problems with many applications.

Another problem with the client/server structure is that the client had to authenticate itself with the database server. To do so, the application required a username and password. This is a security violation if the client software is to reside outside the computing infrastructure of the company. This also posed a problem if the company had a limited number of licences for the database server software. Each client would require a licence in order to connect and the company would soon run out if there were too many clients.

The solution was middleware. Middleware is a piece of software which manages the connection pooling to the database and provides a consolidated interface for client software. This solution gets around the licencing models set by the database manufacturers (it should be noted that many licencing agreements ' strictly forbid this type of architecture. Check your licencing agreement to be sure).

The 3 tier model

As web technologies were developed, it quickly became apparent that the client/server model is not appropriate for Internet based applications (based largely on the problems identified above). The solution is a three tier model which includes a client, an application server (which houses the application logic) and a database server (which houses the data management logic). Figure 4 illustrates this architecture.



Figure 4: Three tier model with Application logic and Data Management logic deployed on

2 separate servers

The attraction of this architecture is that many client processes can be supported by a relatively small number of application servers. Similarly, application servers can be redundantly deployed for load balancing and fault tolerance purpose. Many database applications also allow for redundant deployment for load balancing and fault tolerance (see figure 5).

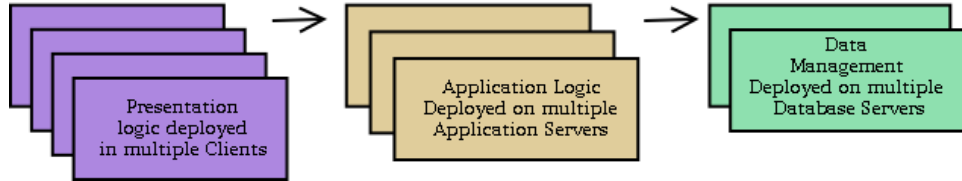
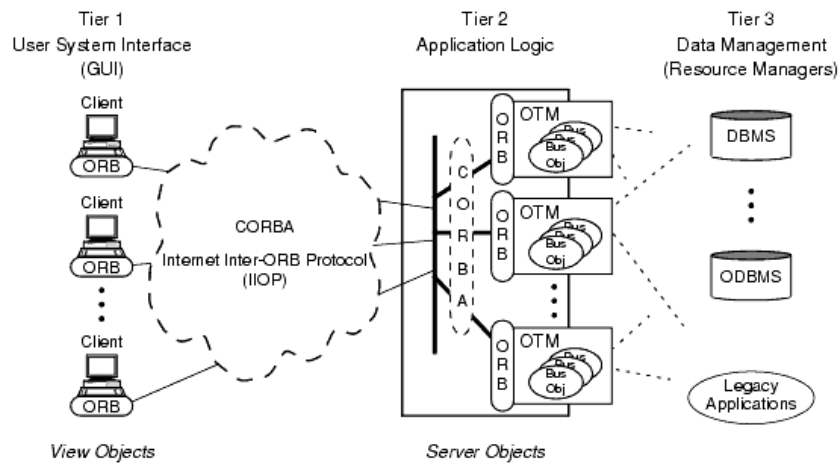


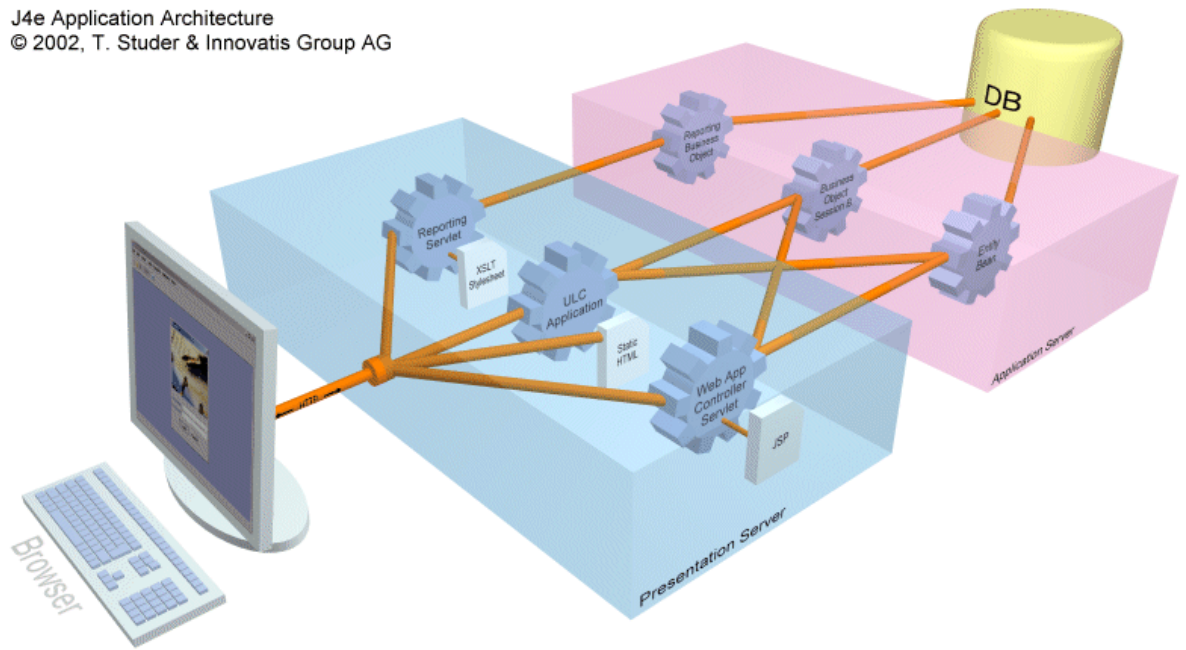
Figure 5: 3 Tier application deployed on multiple application and database servers



Layers within Layers For the purposes of providing detail within the Software Architecture, each of the 3 Layers (Presentation, Application and Data Management) can be broken down into sub-layers. The purpose of creating sublayers is to define the responsibilities (cohesion) of each layer as well as the method of communication between adjacent layers. Figure 6 illustrates sublayers for a web based application using JSP and servlets and an oracle database.

Presentation		Application		Data Management		
Web Browser/HTML Rendering	JSP/HTML Generation	Servlet Layer/ Web Request to Application Mapping	Application Layer Sub Components	Data Access Layer/ SQL Generation	Database Connectivity/ Connection Pooling & Database Mediation	Database/ Storing and Retrieving Data

Figure 6: Web-based application and sub-layers



Once the various layers and their respective responsibilities have been defined, the System Architect can then identify the method and protocol of communication between the various layers (Figure 7).

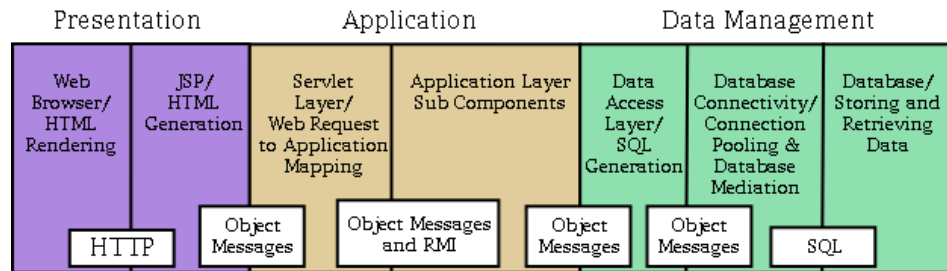


Figure 7: Web-based application and sub-layers and communication infrastructure