

What is Middleware

From Wikipedia:

Middleware is a computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue".^[1] Middleware makes it easier for software developers to perform communication and input/output, so they can focus on the specific purpose of their application. Middleware is the software that connects software components or enterprise applications. Middleware is the software layer that lies between the operating system and the applications on each side of a distributed computer network (Figure 1-1). Typically, it supports complex, distributed business software applications.

From Oracle

1.1 What is Middleware?

Middleware is the software that connects software components or enterprise applications. Middleware is the software layer that lies between the operating system and the applications on each side of a distributed computer network (Figure 1-1). Typically, it supports complex, distributed business software applications.

Middleware is the infrastructure which facilitates creation of business applications, and provides core services like concurrency, transactions, threading, messaging, and the SCA framework for service-oriented architecture (SOA) applications. It also provides security and enables high availability functionality to your enterprise.

Middleware includes Web servers, application servers, content management systems, and similar tools that support application development and delivery. It is especially integral to information technology based on Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web services, SOA, Web 2.0 infrastructure, and Lightweight Directory Access Protocol (LDAP) etc.

When

- Middleware became a popular term in the 90's
- It was first used in 1968 in report of the 1968 NATO Software Engineering Conference (http://ironick.typepad.com/ironick/2005/07/update_on_the_o.html)
- Works related to middleware happened in 1980's: research projects about distributed objects created middleware to support these objects
- Still viable
- Trend: to handle mobile app, cloud computing, business processing management applications...

Why

- Business level
 - Agile: develop products or systems quicker
 - Increased efficiency, which can reduce business costs
 - Innovation capabilities: which is a competitive advantage over competitors
- Technical level
 - Reduce heterogeneous complexities
 - Reuse legacy systems
 - to bring together resources across dissimilar networks or computing platforms.

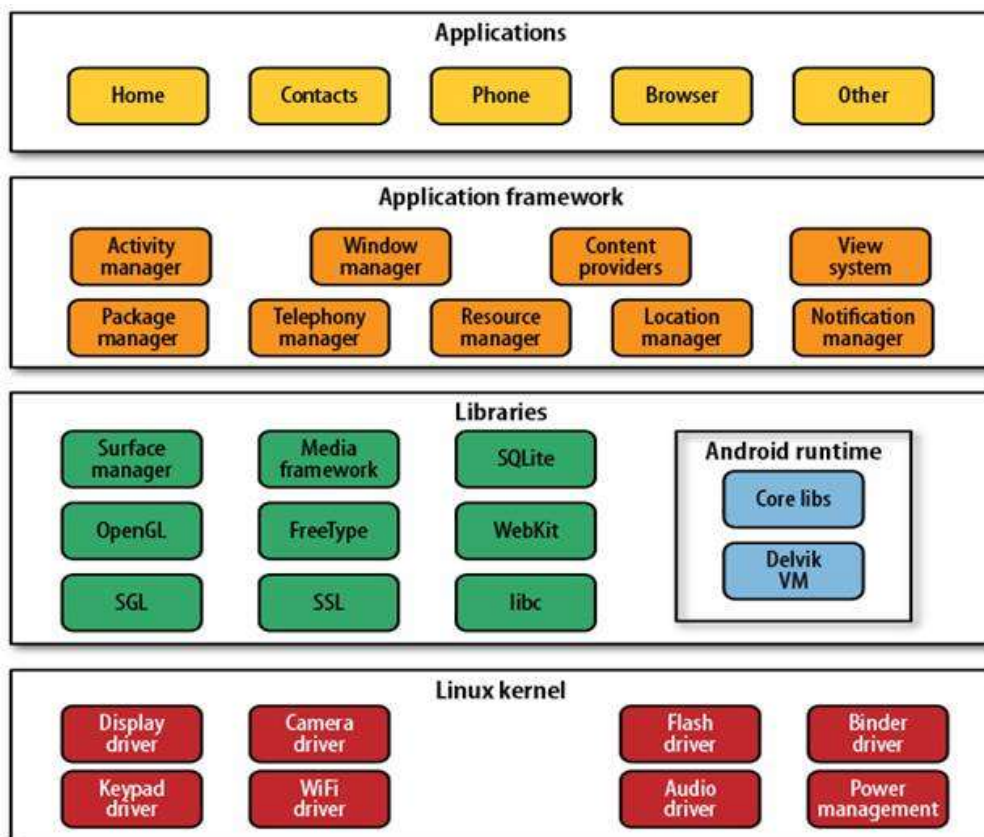
- Prevent vendor lock-in
- Independent from network types, and operation systems
- System/software availability, reliability, and scalability

Examples of Middleware

Examples 1 Android

In between Linux OS and android apps

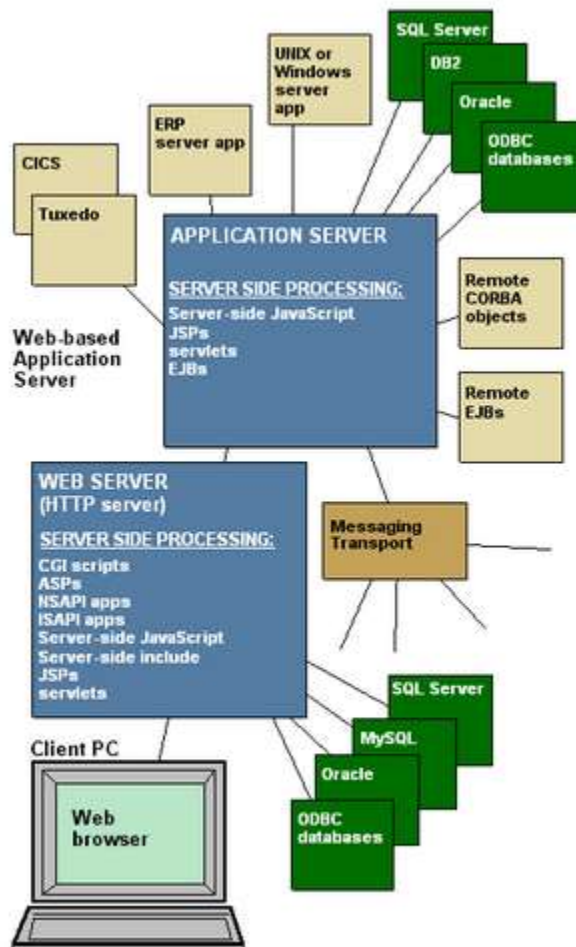
The [Android](#) operating system uses the [Linux](#) kernel at its core, and also provides an [application framework](#) that developers incorporate into their applications. In addition, [Android](#) provides a middleware layer including [libraries](#) that provide services such as data storage, screen display, [multimedia](#), and web browsing. Because the middleware libraries are [compiled](#) to [machine language](#), services execute quickly. Middleware libraries also implement device-specific functions, so applications and the application framework need not concern themselves with variations between various Android devices. Android's middleware layer also contains the [Dalvik virtual machine](#) and its core Java application libraries.^[3]



Example 2: Game engines

Game engine software such as [Gamebryo](#) and [Renderware](#) are sometimes described as middleware, because they provide many services to simplify game development.

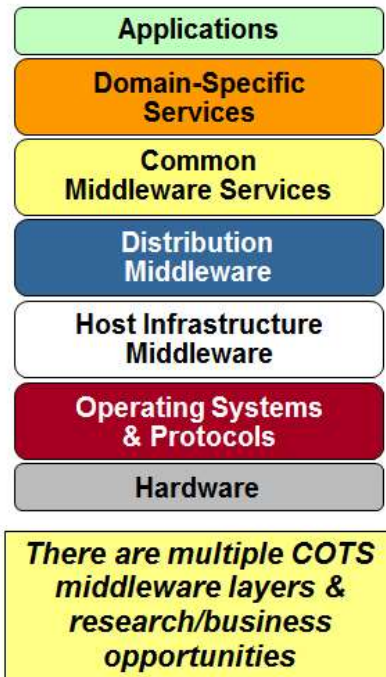
Example 3: application servers – like Redhat JBOSS or IBM Websphere



Example 4:



The Evolution of Middleware



Historically, mission-critical apps were built directly atop hardware & OS

- Tedious, error-prone, & costly over lifecycles

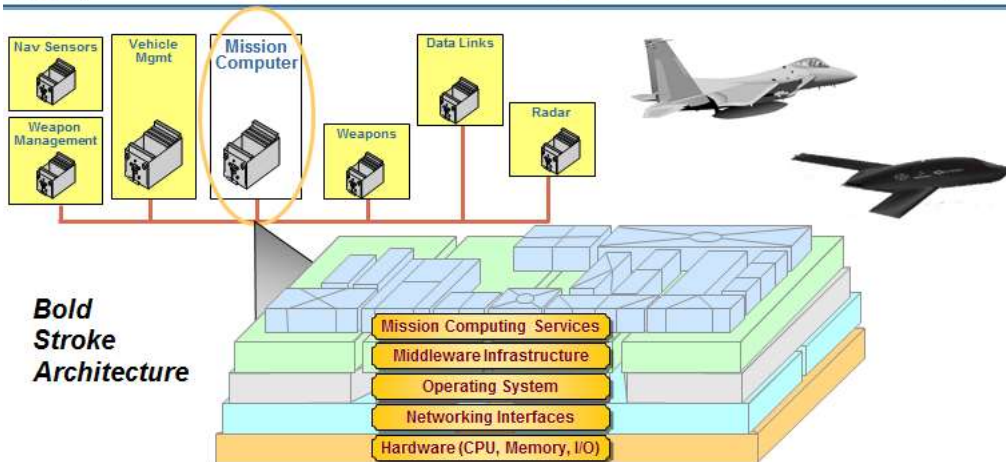
There are layers of middleware, just like there are layers of networking protocols

Standards-based COTS middleware helps:

- Control end-to-end resources & QoS
- Leverage hardware & software technology advances
- Evolve to new environments & requirements
- Provide a wide array of reusable, off-the-shelf developer-oriented services



Example: Boeing Bold Stroke

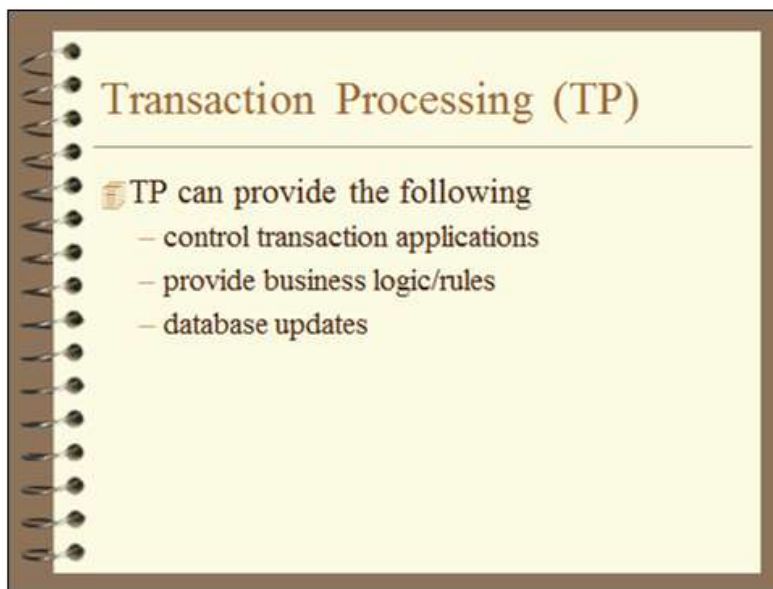
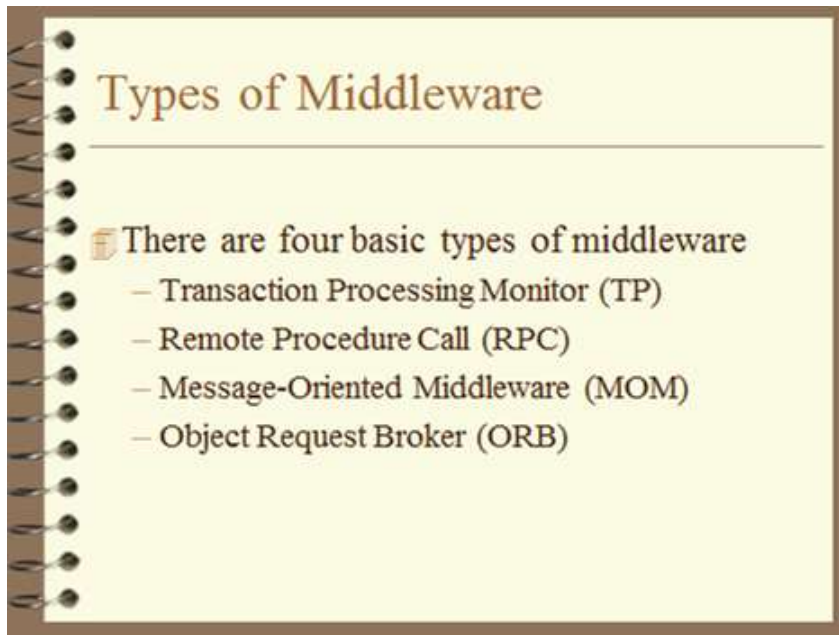


- Avionics mission computing product-line architecture for Boeing military aircraft, e.g., F-18 E/F, 15E, Harrier, UCAV
- DRE system with 100+ developers, 3,000+ software components, 3-5 million lines of C++ code

- Based on COTS hardware, networks, operating systems, & middleware
- Used as Open Experimentation Platform (OEP) for DARPA IXO PCES, MoBIES, SEC, MICA programs

Type of Middleware

4 types that we want to focus related to impact to Software Architecture



TP monitor technology is not a new idea. This model has been used for many years in areas such as data management, network access, and customer service, to name a few. This is your basic three tier architecture with a fancy name within the context frame of middleware.

Remote Procedure Call (RPC)

- ☞ RPC is a client/server mechanism that allows the program to be **distributed** across multiple platforms.
- ☞ RPC's reduce the complexity of a system that spans multiple operating systems and network protocols by **hiding OS and network interface details** from the programmer.

RPC is also a very old technology. The first full scale implementations date back to the late 1970's. RPC provides a means to make the invocation of a routine on a server by a client application on a different machine look like any function invocation within the client application (well almost, issues such as marshalling data structures into serialized representations make total transparency impossible). This is done by embedding special function calls, RPC's into the client or server application. Because these special calls are embedded, RPC's do not constitute stand alone middleware

Portability is increased by **minimizing** the amount of **client code** and making network communications transparent to the application. This makes it possible to port the code to a new platform without having to change network or OS system calls

The flexibility of architecture increases by allowing client applications to **invoke remote server components** without ever knowing what network and OS API's are used.

Remote Procedure Call (cont'd)

RPC's are usually implemented by:

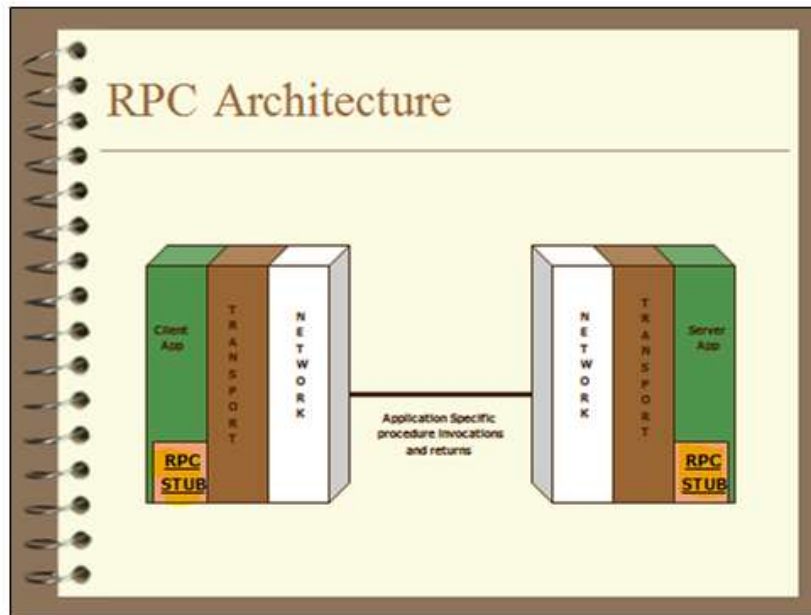
- proprietary products
- proprietary development tools that create client server stubs

Most RPC implementations use a **synchronous** (call/wait) protocol to communicate between client and server.

Since there is no standard for RPC implementation, all RPC development environments are the result of private proprietary systems.


RPC compilers are used to create client/server stubs, the embedded code that implements OS and network interface specific functions, hiding these details from the developer.

The problem with **synchronous communication** is that it is a **blocking protocol**. For example, if a client makes a request to the server, the **client blocks until it receives a reply from the server**.



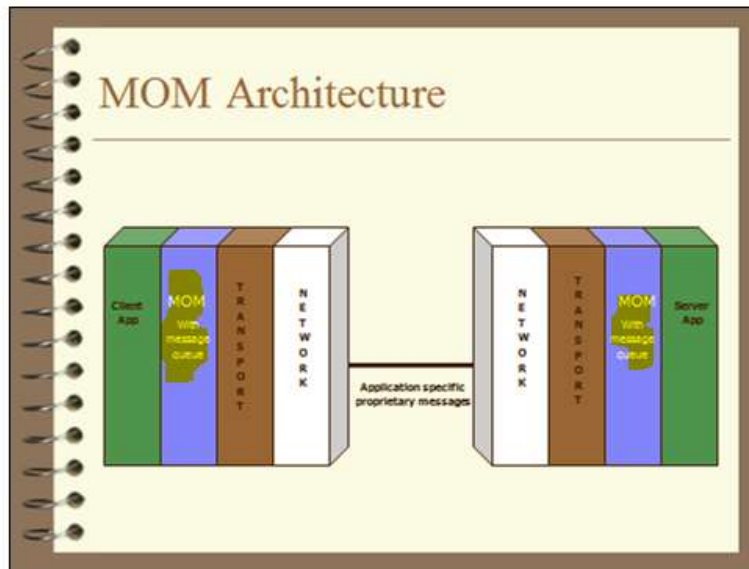
The RPC compiler for a given operating system provides client/server stubs which are interfaces to the operating systems transport layer, which is in turn an interface to the network layer implementation to the system. For this reason, the application developer does not need to worry about OS specific system calls.

Message Oriented Middleware (MOM)

 Message-oriented middleware

- functionality similar to RPC
- provides asynchronous communication between client and server applications by queueing messages temporarily when one or the other is busy or not connected.

MOM architecture has one-upped RPC by providing a queuing mechanism. This is simply a means to accomplish asynchronous communication by saving messages for target nodes when they are busy, slow, or even dead.



MOM makes asynchronous communication possible by queueing messages between the transport and application layers so that client and server processes do not have to sit around waiting on each other. Isn't that nice of her?

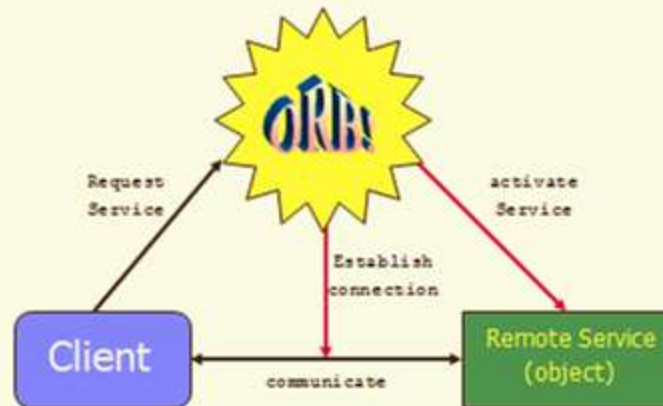
Object Request Broker (ORB)

- ORB's overcome; machine, software and vendors are no boundaries for the ORB.
- Major functionality includes:
 - interface definition
 - location and activation of remote objects
 - communication between clients and objects

I like to think of an object request broker as a name service for objects. The server process registers its service with the ORB. The client contacts the ORB asking where it can find the service (resolving the location). But then the ORB takes us **one step further**, and actually sets up the client-server connection.

The best part is that the developer never has to worry about any of this. To the programmer, the call looks like it is local to the client.

ORB Architecture

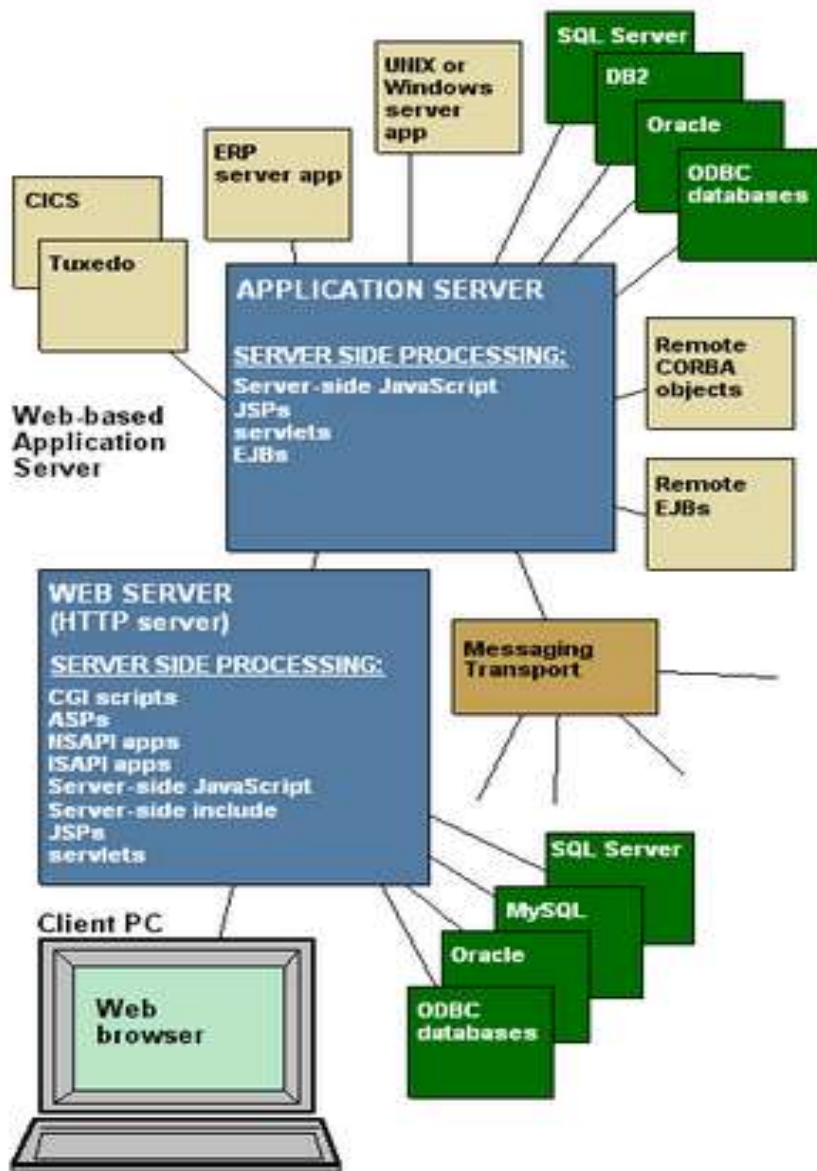


The client makes a request to the ORB to find and bind to the needed service. The ORB finds the service, establishes a connection between the client and server processes and finally activates the service.

The ORB allows the server to hide implementation details from the client. The client never has to know what programming language was used, what operating system the process is running on, what hardware architecture the host has, or even the location of the object.

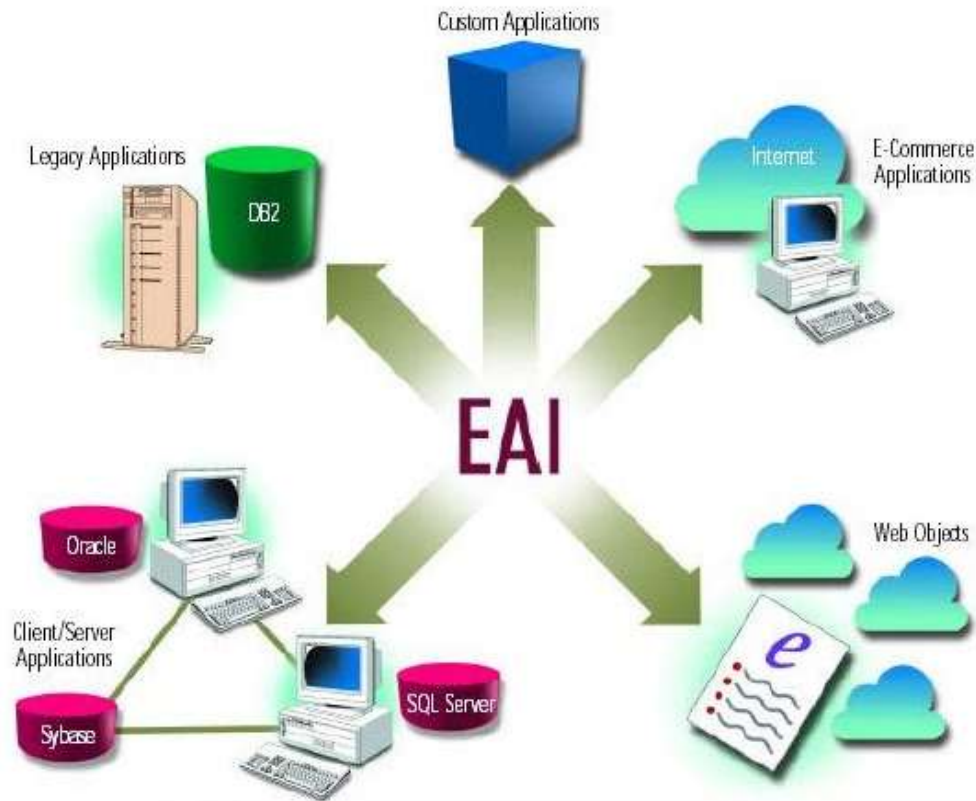
Other types of Middleware are

Application Server



Enterprise Application Integration (EAI)

EAI (Enterprise Application Integration) is a middleware technology that EAI provides. It is a medium for systems to talk and act together and use their common feature to each other.



Read more

- <http://condor.depaul.edu/elliott/513/projects-archive/DS513Fall98/seminar/Plantain/middleware.ppt>
- <http://sardes.inrialpes.fr/~krakowia/MW-Book/main-onebib.pdf>
- <http://www.pcmag.com/encyclopedia/term/47013/middleware>
- https://en.wikipedia.org/wiki/Enterprise_application_integration#Technologies
- <http://www.edisourcing.com/Services/BusinessIntegration/EnterpriseApplicationIntegrationEAI.aspx>

Review Questions

Section 1 - True/False Question

- 1- (T/F) - Middleware allows a company to develop software products or systems quicker
- 2- (T/F) - Middleware help company to achieve efficiency, which can reduce business costs
- 3- (T/F) - Company achieves innovation with middleware solution, which is a competitive advantage over competitors
- 4- (T/F) - Middleware became popular during 2000's
- 5- (T/F) - An example of middleware is the Android libraries that provide services such as data storage, multimedia, screen display and web browsing.
- 6- (T/F) - Android application framework is an example of middleware
- 7- (T/F) - Another example of middleware is application servers like IBM WebSphere
- 8- (T/F) - Most RPC implementations use synchronous protocol to communicate between client and server
- 9- (T/F) - Message Oriented Middleware is similar to RPC and provides synchronous communication between client and server applications
- 10- ORB is the older technology compared with RPC

Section 2 - MCQ

- 11- Which of the following is not a type of middleware
 - a) Transaction Processing Monitor
 - b) Remote Procedure Call
 - c) Message Oriented Middleware
 - d) Object Request Broker
 - e) Operating System Kernel
- 12- What is not a technology benefit of middleware
 - a) Reduce heterogeneous complexities
 - b) Reuse legacy systems
 - c) Expensive to develop
 - d) to bring together resources across dissimilar networks or computing platforms.
- 13- What is RPC?
 - a) Remote Procedure Call
 - b) Report Processing Connectivity
 - c) Remit Protocol Control
 - d) None of the above

Answer

Section 1 - True/False Questions

1-T

2-T

3-T

4-F

5-T

6-F

7- T

8-T

9-F (it uses asynchronous communication)

10-F (RPC is the oldest technology compared to MOM and ORB)

Section 2 – Multiple Choice Questions

11- e

12- c

13- a