Experiment No: 3

**Aim:** Case Study Of CORBA

**Middleware:**

Middleware is the software that connects network-based requests generated by a client to the back-end data the client is requesting. It is a general term for software that serves to "glue together" separate, often complex and already existing programs.

With network-based interactions, a client, or requesting program, can make a request. That client is typically an application that resides on the front end, which is where the user interacts with software. Resources such as databases, message queues, NoSQL data stores and file servers are often referred to as being part of the back end.

The name middleware stems from the fact that it is the software that sits between the client-side request on the front end and the back-end resource being requested.

**Introduction to CORBA:**

The Common Object Request Broker Architecture (CORBA) is a specification developed by the Object Management Group (OMG). CORBA describes a messaging mechanism by which objects distributed over a network can communicate with each other irrespective of the platform and language used to develop those objects.

There are two basic types of objects in CORBA.

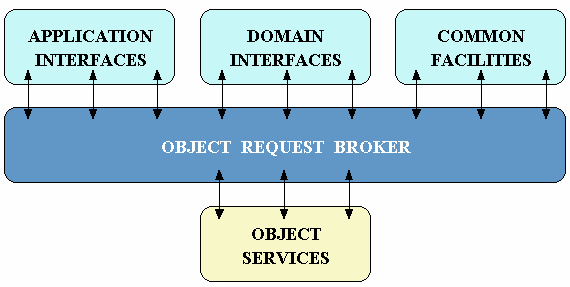
* The object that includes some functionality and may be used by other objects is called a service provider.
* The object that requires the services of other objects is called the client.

The service provider object and client object communicate with each other independent of the programming language used to design them and independent of the operating system in which they run. Each service provider defines an interface, which provides a description of the services provided by the client.

CORBA enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services. More specifically, CORBA is a mechanism in software for normalizing the method-call semantics between application objects residing either in the same address space (application) or remote address space (same host, or remote host on a network).

CORBA applications are composed of objects that combine data and functions that represent something in the real world. Each object has multiple instances, and each instance is associated with a particular client request. For example, a bank teller object has multiple instances, each of which is specific to an individual customer. Each object indicates all the services it provides, the input essential for each service and the output of a service, if any, in the form of a file in a language known as the Interface Definition Language (IDL). The client object that is seeking to access a specific operation on the object uses the IDL file to see the available services and marshal the arguments appropriately.

**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 are 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. Other OMG RFPs will soon be issued in the telecommunications, medical, and financial domains.
* **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.

**Features:**

* **Objects By Reference:** This reference is either acquired through a stringified Uniform Resource Locator (URL), NameService lookup (similar to Domain Name System (DNS)), or passed-in as a method parameter during a call. Object references are lightweight objects matching the interface of the real object (remote or local). Method calls on the reference result in subsequent calls to the ORB and blocking on the thread while waiting for a reply, success or failure. The parameters, return data (if any), and exception data are marshaled internally by the ORB according to the local language and OS mapping.
* **Data By Value:** The CORBA Interface Definition Language provides the language- and OS-neutral inter-object communication definition. CORBA Objects are passed by reference, while data (integers, doubles, structs, enums, etc.) are passed by value. The combination of Objects-by-reference and data-by-value provides the means to enforce strong data typing while compiling clients and servers, yet preserve the flexibility inherent in the CORBA problem-space.
* **Objects By Value (OBV):** Apart from remote objects, the CORBA and RMI-IIOP define the concept of the OBV and Valuetypes. The code inside the methods of Valuetype objects is executed locally by default. If the OBV has been received from the remote side, the needed code must be either a priori known for both sides or dynamically downloaded from the sender. To make this possible, the record, defining OBV, contains the Code Base that is a space-separated list of URLs whence this code should be downloaded. The OBV can also have the remote methods.
* **CORBA Component Model (CCM):** CORBA Component Model (CCM) is an addition to the family of CORBA definitions.It was introduced with CORBA 3 and it describes a standard application framework for CORBA components. Though not dependent on "language dependent Enterprise Java Beans (EJB)", it is a more general form of EJB, providing four component types instead of the two that EJB defines. It provides an abstraction of entities that can provide and accept services through well-defined named interfaces called ports. The CCM has a component container, where software components can be deployed. The container offers a set of services that the components can use. These services include (but are not limited to) notification, authentication, persistence and transaction processing. These are the most-used services any distributed system requires, and, by moving the implementation of these services from the software components to the component container, the complexity of the components is dramatically reduced.
* **Portable interceptors:** Portable interceptors are the "hooks", used by CORBA and RMI-IIOP to mediate the most important functions of the CORBA system. The CORBA standard defines the following types of interceptors:
* IOR interceptors mediate the creation of the new references to the remote objects, presented by the current server.
* Client interceptors usually mediate the remote method calls on the client (caller) side. If the object Servant exists on the same server where the method is invoked, they also mediate the local calls.
* Server interceptors mediate the handling of the remote method calls on the server (handler) side.

The interceptors can attach the specific information to the messages being sent and IORs being created. This information can be later read by the corresponding interceptor on the remote side. Interceptors can also throw forwarding exceptions, redirecting request to another target.

**Applications**

CORBA used with Java can provide perfect *Internet/Intranet* Solutions.

1. Java applets can be downloaded via web based applications. These Java applets are capable of directly accessing CORBA objects via IIOP. There are a number of Java based ORBs available on the market. By introducing CORBA communication into a Java applet, arbitrary CORBA services can be accessed directly. These services can be developed in any language supported by CORBA or on top of any CORBA product that supports IIOP.
2. Pure HTML based application are capable of accessing CORBA objects via CGI gateways. Arbitrary unknown CORBA objects can be accessed by a single pre-compiled client application via Dynamic Invocation. A pre-compiled application can dynamically generate HTML pages based upon results obtained from arbitrary invocation of operations. This solution has the advantage of being based only upon HTML, it is not specific to a particular web browser.
3. A plug-in can be developed for a particular browser which enables it to speak directly to any CORBA object through IIOP.
4. Web servers from Netscape and Oracle are beginning to support IIOP directly. This means that in addition to supporting HTTP, FTP access and news group access, they will be capable of accessing any CORBA object capable of supporting IIOP.

*Software Products*

1. Presently CORBA based application development prevails in all type of Industries that includes Aerospace/Defense, Banking/Finance, Chemical/Petrochemical, Electronic Commerce, Government, Healthcare/Insurance, Manufacturing, Publishing/Multimedia , Telecommunications.
2. CORBA is also playing important role in software product development.
3. Netscape is making CORBA ubiquitous on the client. It is bundling the VisiBroker for Java ORB with every browser. Netscape is also using CORBA for its server-to-server infrastructure.
4. Oracle has adopted CORBA as the platform for its Network Computing Architecture. Oracle's entire software line, from the database engines to stored procedures, tools, and the Internet, will be built on a CORBA object bus. For example, the database engine will be componentized using CORBA. Third parties will be able extend the database using CORBA components called Cartridges. Oracle is building most of the CORBA Services on top of the Visigenic IIOP ORB. This ORB will first appear in the next release of Oracle Web Server; it will serve as the foundation for Oracle's Internet products.
5. JavaSoft is making CORBA the foundation for distributed Java. SunSoft is building its Internet server strategy around CORBA using its NEO ORB and Solstice.
6. IBM/Lotus is building its cross-platform network computing infrastructure on CORBA/Java. IBM intends to bundle a Java runtime with all its OS platforms

**Advantages of CORBA:**

* Multilingual
* Multiplatform
* Relatively lightweight
* Object Location Transparency
* Server Transparency
* Operating System Transparency
* Protocol Transparency
* Language Transparency
* Architecture Transparency
* Good for infrastructure application
  + Like management application

**Disadvantages of CORBA:**

* Integration required
* Programming model in industry
* Not primary
* Two primary vendors: Borland and Iona
* Less control over how your information is actually sent

**Conclusion:** Thus, we have successfully studied CORBA.