Chapter – 8

Emerging Trends

Client Server Software:

A client is basically a consumer of services and server is a provider of services. A client requests some services from the server and the server provides the required services to the client. Client and server are usually software components running on independent machines. Even a single machine can sometimes acts as a client and at other times a server depending on the situations. Thus, client and server are mere roles.

Example:

A man was visiting his friend's town in his car. The man had a handheld computer (client). He knew his friend's name but he didn't know his friend's address. So he sent a wireless message (request) to the nearest "address server" by his handheld computer to enquire his friend's address. The message first came to the base station. The base station forwarded that message through landline to local area network where the server is located. After some processing, LAN sent back that friend's address (service) to the man.

Advantages of client-server software development

There are many advantages of client-server software products as compared to monolithic ones. These advantages are:

- ☑ **Simplicity And Modularity:**

Client and server components are loosely coupled and therefore modular. These are easy to understand and develop.

- ☑ **Flexibility:**

Both client and server software can be easily migrated across different machines in case some machine becomes unavailable or crashes. The client can access the service anywhere. Also, clients and servers can be added incrementally.

- ☑ **Extensibility:**

More servers and clients can be effortlessly added.

- ☑ **Concurrency:**

The processing is naturally divided across several machines. Clients and servers reside in different machines which can operate in parallel and thus processing becomes faster.

- ☑ **Cost Effectiveness:**

Clients can be cheap desktop computers whereas severs can be sophisticated and expensive computers. To use a sophisticated software, one needs to own only a cheap client and invoke the server.

☑ **Specialization:**

One can have different types of computers to run different types of servers. Thus, servers can be specialized to solve some specific problems.

☑ **Current trend:**

Mobile computing implicitly uses client-server technique. Cell phones (handheld computers) are being provided with small processing power, keyboard, small memory, and LCD display. Cell phones cannot really compute much as they have very limited processing power and storage capacity but they can act as clients. The handhold computers only support the interface to place requests on some remote servers.

☑ **Application Service Providers (ASPs):**

There are many application software products which are very expensive. Thus it makes prohibitively costly to own those applications. The cost of those applications often runs into millions of dollars. For example, a Chemical Simulation Software named "Aspen" is very expensive but very powerful. For small industries it would not be practical to own that software. Application Service Providers can own ASPEN and let the small industries use it as client and charge them based on usage time. A client and simply logs in and ASP charges according to the time that the software is used.

☑ **Component-Based Development:**

It is the enabler of the client-server technology. Component-based development is radically different from traditional software development. In component-based development, a developer essentially integrates pre-built components purchased off-the-shelf. This is akin to the way hardware developers integrate ICs on a Printed Circuit Board (PCB). Components might reside on different computers which act as servers and clients.

☑ **Fault Tolerance:**

Client-server based systems are usually fault-tolerant. There can be many servers. If one server crashes then client requests can be switched to a redundant server.

Disadvantages of client-server software

There are several disadvantages of client-server software development. Those disadvantages are:

☑ **Security:**

In a monolithic application, implementation of security is very easy. But in a client-server based development a lot of flexibility is provided and a client can connect from anywhere. This makes it easy for hackers to break into the system. Therefore, ensuring security in client-server system is very challenging.

☑ **Servers Can Be Bottlenecks:**

Servers can turn out to be bottlenecks because many clients might try to connect to a server at the same time. This problem arises due to the flexibility given that any client can connect anytime required.

☑ **Compatibility:**

Clients and servers may not be compatible to each other. Since the client and server components may be manufactured by different vendors, they may not be compatible with respect to data types, language, etc.

☑ **Inconsistency:**

Replication of servers is a problem as it can make data inconsistent.

CORBA:

Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group (OMG), which currently includes over 500 member companies. Both International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components). CORBA 3 is the latest level.

The essential concept in CORBA is the Object Request Broker (ORB). ORB support in a network of clients and servers on different computers means that a client program (which may itself be an object) can request services from a server program or object without having to understand where the server is in a distributed network or what the interface to the server program looks like. To make requests or return replies between the ORBs, programs use the General Inter-ORB Protocol (GIOP) and, for the Internet, it's Internet Inter-ORB Protocol (IIOP). IIOP maps GIOP requests and replies to the Internet's Transmission Control Protocol (TCP) layer in each computer.

A notable hold-out from CORBA is Microsoft, which has its own distributed object architecture, the Distributed Component Object Model (DCOM). However, CORBA and Microsoft have agreed on a gateway approach so that a client object developed with the Component Object Model will be able to communicate with a CORBA server (and vice versa).

Distributed Computing Environment (DCE), a distributed programming architecture that preceded the trend toward object-oriented programming and CORBA, is currently used by a number of large companies. DCE will perhaps continue to exist along with CORBA and there will be "bridges" between the two.

COM (COMPONENT OBJECT MODEL):

Component Object Model (COM) is a binary-interface standard for software components introduced by Microsoft in 1993. It is used to enable inter-process communication and dynamic object creation in a large range of programming languages. COM is the basis for several other Microsoft technologies and frameworks, including OLE, OLE Automation, Browser Helper

Object, ActiveX, COM+, DCOM, the Windows shell, DirectX, UMDF and Windows Runtime. The essence of COM is a language-neutral way of implementing objects that can be used in environments different from the one in which they were created, even across machine boundaries. For well-authored components, COM allows reuse of objects with no knowledge of their internal implementation, as it forces component implementers to provide well-defined interfaces that are separated from the implementation. The different allocation semantics of languages are accommodated by making objects responsible for their own creation and destruction through reference-counting. Type conversion casting between different interfaces of an object is achieved through the Query Interface method. The preferred method of "inheritance" within COM is the creation of sub-objects to which method "calls" are delegated.

DCOM (Distributed Component Object Model):

Distributed Component Object Model (DCOM) is a proprietary Microsoft technology for communication between software components on networked computers. DCOM, which originally was called "Network OLE", extends Microsoft's COM, and provides the communication substrate under Microsoft's COM+ application server infrastructure.

The addition of the "D" to COM was due to extensive use of DCE/RPC (Distributed Computing Environment/Remote Procedure Calls) more specifically Microsoft's enhanced version, known as MSRPC.

In terms of the extensions it added to COM, DCOM had to solve the problems of:

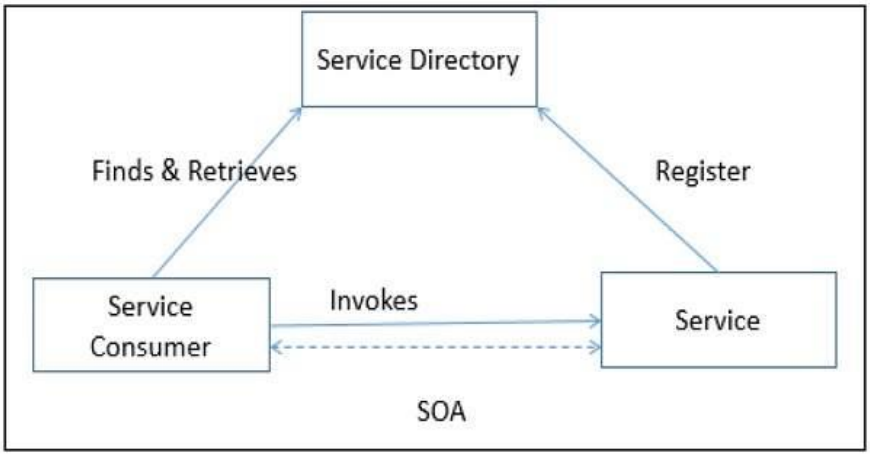
- ☑ Marshalling: serializing and deserializing the arguments and return values of method calls "over the wire".
- ☑ Distributed garbage collection: ensuring that references held by clients of interfaces are released when, for example, the client process crashed, or the network connection was lost.
- ☑ It had to combine Hundreds/Tens of Thousands of objects held in the client's browser with a single transmission in order to minimize bandwidth utilization.

DCOM was a major competitor to CORBA.

Service-Oriented Architecture (SOA):

A service is a component of business functionality that is well-defined, self-contained, independent, published, and available to be used via a standard programming interface. The connections between services are conducted by common and universal message-oriented protocols such as the SOAP Web service protocol, which can deliver requests and responses between services loosely.

Service-oriented architecture is a client/server design which support business-driven IT approach in which an application consists of software services and software service consumers (also known as clients or service requesters).



Features of SOA:

☑ **Distributed Deployment**:

Expose enterprise data and business logic as loosely, coupled, discoverable, structured, standard-based, coarse-grained, stateless units of functionality called services.

☑ **Composability**:

Assemble new processes from existing services that are exposed at a desired granularity through well defined, published, and standard complaint interfaces.

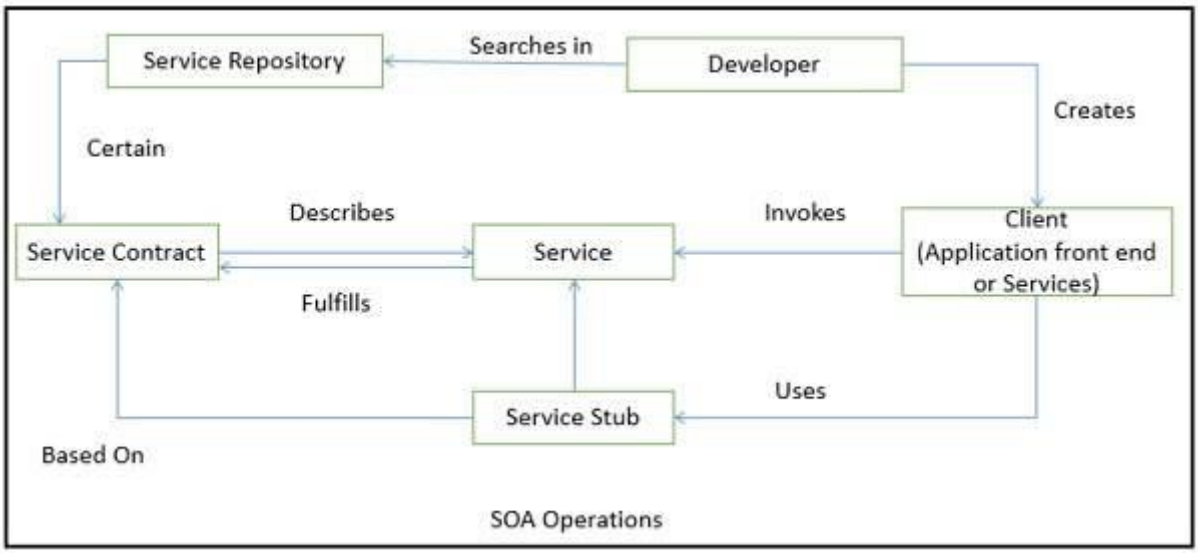
☑ **Interoperability**:

Share capabilities and reuse shared services across a network irrespective of underlying protocols or implementation technology.

☑ **Reusability**:

Choose a service provider and access to existing resources exposed as services.

SOA Operation:



SAAS (SOFTWARE AS A SERVICE):

Software as a service (SaaS) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. SaaS is one of three main categories of cloud computing, alongside infrastructure as a service (IaaS) and platform as a service (PaaS).

SaaS removes the need for organizations to install and run applications on their own computers or in their own data centers. This eliminates the expense of hardware acquisition, provisioning and maintenance, as well as software licensing, installation and support. Other benefits of the SaaS model include:

1. Flexible payments:

Rather than purchasing software to install, or additional hardware to support it, customers subscribe to a SaaS offering. Generally, they pay for this service on a monthly basis using a pay-as-you-go model. Transitioning costs to a recurring operating expense allows many businesses to exercise better and more predictable budgeting. Users can also terminate SaaS offerings at any time to stop those recurring costs.

2. Scalable usage:

Cloud services like SaaS offer high scalability, which gives customers the option to access more, or fewer, services or features on-demand.

3. Automatic updates:

Rather than purchasing new software, customers can rely on a SaaS provider to automatically perform updates and patch management. This further reduces the burden on in-house IT staff.

4. Accessibility and persistence:

Since SaaS applications are delivered over the Internet, users can access them from any Internet-enabled device and location.