SERVICE ORIENTED ARCHITECTURE

IMPORTANCE OF SOA

Service Oriented Architecture (SOA) describes a standard method for requesting services from distributed components and managing the results.

Because the clients requesting services, the components providing the services, the protocols used to deliver messages, and the responses can vary widely, SOA provides the translation and management layer in an architecture that removes the barrier for a client obtaining desired services.

With SOA, clients and components can be written in different languages and can use multiple messaging protocols and networking protocols to communicate with one another.

SOA provides the standards that transport the messages and makes the infrastructure to support it possible.

SOA provides access to reusable Web services over a TCP/IP network, which makes this an important topic to cloud computing going forward.

You don't need SOA if you are creating a monolithic cloud application that performs a specific function such as backup, e-mail, Web page access, or instant messaging.

SOA offers access to ready-made, modular, highly optimized, and widely shareable components that can minimize developer and infrastructure costs.

HORIZONTAL AND VERTICAL SCALING

Whereas SOA can be used to construct large and complex applications that scale both horizontally and vertically, cloud computing applications tend to be scaled vertically.

Horizontal scaling refers to applications with a large number of different business processes operating. Vertical scaling refers to large applications with a limited number of business processes operating.

SOA techniques may be applied in both instances.

INTRODUCING SOA

Service Oriented Architecture (SOA) is a specification and a methodology for providing platform and language-independent services for use in distributed applications.

A service is a repeatable task within a business process, and a business task is a composition of services. SOA describes a message- passing taxonomy for a component-based architecture that provides services to clients upon demand.

Clients access a component that complies with SOA by passing a message containing metadata to be acted upon in a standard format.

The component acts on that message and returns a response that the client then uses for its own purpose.

A common example of a message is an XML file transported over a network protocol such as SOAP.

Usually service providers and service consumers do not pass messages directly to each other.

Implementations of SOA employ middleware software to play the role of transaction manager (or broker) and translator.

That middleware can discover and list available services, as well as potential service consumers, often in the form of a registry, because SOA describes a distributed architecture security and trust services are built directly into many of these products to protect communication.

Middleware products also can be where the logic of business processes reside; they can be general purpose applications, industry-specific, private, or public services.

Middleware services manage lookup requests.

The Universal Description Discovery and Integration (UDDI) protocol is the one most commonly used to broadcast and discover available Web services, often passing data in the form of an Electronic Business using eXtensible Markup Language (ebXML) documents.

Service consumers find a Web service in a broker registry and bind their service requests to that specific service; if the broker supports several Web services, it can bind to any of the ones that are useful.

This architecture does not contain executable links that require access to a specific API.

The message presents data to the service, and the service responds. It is up to the client to determine if the service returned an appropriate result.

An SOA is then seen as a method for creating an integrated process as a set of linked services.

The component exposes itself as an “endpoint” (a term of art in SOA) to the client.

PROTOCOL STACK FOR SOA

The most commonly used message-passing format is an Extensible Markup Language (XML) document using Simple Object Access Protocol (SOAP), but many more are possible, including Web Services Description Language (WSDL), Web Services Security (WSS), and Business Process Execution Language for Web Services (WS-BPEL).

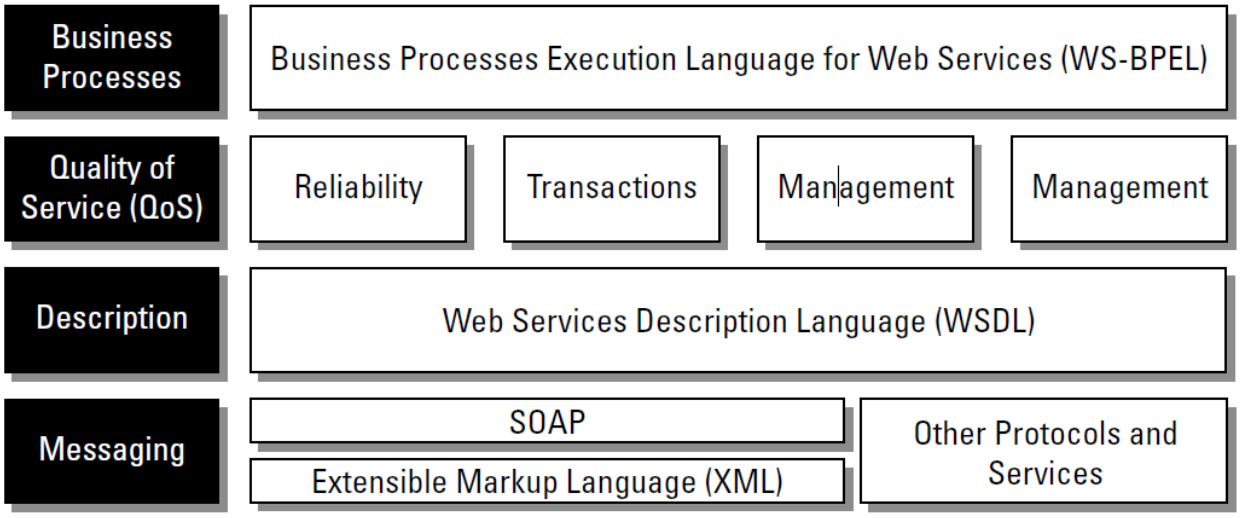
WSDL is commonly used to describe the service interface, how to bind information, and the nature of the component's service or endpoint.

The Service Component Definition Language (SCDL) is used to define the service component that performs the service, providing the component service information that is not part of the Web service and that therefore wouldn't be part of WSDL.

Figure 13.1 shows a protocol stack for SOA architecture and how those different protocols execute the functions required in the Service Oriented Architecture.

In the figure, the box labeled Other Services could include Common Object Request Broker Architecture (CORBA), Representational State Transfer (REST), Remote Procedure Calls (RPC), Distributed Common Object Model (DCOM), Jini, Data Distribution Service (DDS), Windows Communication Foundation (WCF), and other technologies and protocols.

It is this flexibility and neutrality that makes SOA so singularly useful in designing complex applications.



COMMUNICATION BETWEEN CLIENTS AND SERVICES

Components are coded with their service logic and their dependencies, QoS is established, and the service is instantiated. In the SCA model, data and messages are exchanged in a Service Data Object (SDO).

This system of messaging using objects and services is sometimes referred to as a Data Access Service (DAS).

Figure 13.2 shows how components of different types can communicate using different protocols as part of SOA.

SOA allows for different component and client construction, as well as access to each using different protocols.