Service Oriented Architecture (SOA) and Web Service

We have **Objects → Modules → Component → Services**

Web services are not inherently service oriented.

A web service merely exposes a capability that conforms to web services.

CBDI → Component Based Development and Integration

SOA Definition by W3C

According W3C (World Wide Consortium), SOA is defined as **a set of components which can be invoked and whose interface descriptions can be published and discovered**.

A system designed to support interoperable machine to machine interaction over a network. It has an interface described in a format that machine can process.

CBDI definition of SOA

SOA is defined as the policies, practices, frameworks that enable application functionality to be provided and consumed as a set of services published to the service consumer. Services can be invoked, published and discovered and abstracted away from implementation using single standard based interface.

CBDI defines SOA as a style resulting from the use of particular policies, practices and frameworks that deliver services that conform to certain norms.

SOA is not just an architecture of services seen from a technology perspective, but the policies, practices and frameworks by which we ensure the right services are provided and consumed.

Web services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form. Web service provide us with certain architectural characteristics and benefits – specifically platform independence, loose coupling, self-description and discovery.

http://www.theserverside.com/discussions/thread.tss?thread_id=34961

Web services are self-describing services that will perform well defined tasks and can be accessed through the web.

Service Oriented Architecture (SOA) is (roughly) an architecture paradigm that focuses on building systems through the use of different Web Services, integrating them together to make up the whole system.

SOA is an architectural concept. Web Services is a realization of SOA, that leverages XML and common Internet protocols (such as HTTP) to deploy (typically) coarse-grained, discoverable services.

SOAP, in turn, is an implementation of Web Services. XML-RPC is another.

http://searchdatamanagement.techtarget.com/answer/Whats-the-difference-between-SOA-and-Web-services

Think of a Web service as a conversation, and SOA as the telephone system. A Web service is a "call" to an application, a system, or a hub that asks a question, like: "Does this customer already exist?" By definition, a Web service uses the web to communicate its business question.

SOA, on the other hand, is the architectural framework that enables a series of those Web services to occur. You don't deploy "an SOA," rather you deliver these Web services using a SOA framework. Usually when someone talks about deploying SOA, they're really talking about breaking up their business processes into Web services. So now, instead of lots of different systems in your company asking the question, "Does this customer already exist?" to its native database, it can simply use a Web service to call a central location and ask the question.

http://www.oracle.com/technetwork/systems/soa-142870.html

SOA and web services are two different things, but web services are the preferred standards-based way to realize SOA.

Service-Oriented Architecture

SOA is an architectural style for building software applications that use services available in a network such as the web. It promotes loose coupling between software components so that they can be reused. Applications in SOA are built based on services. A service is an implementation of a well-defined business functionality, and such services can then be consumed by clients in different applications or business processes.

SOA allows for the reuse of existing assets where new services can be created from an existing IT infrastructure of systems. In other words, it enables businesses to leverage existing investments by allowing them to reuse existing applications, and promises interoperability between heterogeneous applications and technologies. SOA provides a level of flexibility that wasn't possible before in the sense that:

- Services are software components with well-defined interfaces that are implementation-independent. An important aspect of SOA is the separation of the service interface (the what) from its implementation (the how). Such services are consumed by clients that are not concerned with how these services will execute their requests.
- Services are self-contained (perform predetermined tasks) and loosely coupled (for independence)
- Services can be dynamically discovered
- Composite services can be built from aggregates of other services

SOA uses the *find-bind-execute* paradigm as shown in Figure 1. In this paradigm, service providers register their service in a public registry. This registry is used by consumers to find services that match certain criteria. If the registry has such a service, it provides the consumer with a contract and an endpoint address for that service.

SOA's Find-Bind-Execute Paradigm

SOA-based applications are distributed multi-tier applications that have presentation, business logic, and persistence layers. Services are the building blocks of SOA applications. While any functionality can be made into a service, the challenge is to define a service interface that is at the right level of abstraction. Services should provide coarse-grained functionality.

Realizing SOA with Web Services

Web services are software systems designed to support interoperable machine-to-machine interaction over a network. This interoperability is gained through a set of XML-based open standards, such as WSDL, SOAP, and UDDI. These standards provide a common approach for defining, publishing, and using web services.

Interoperability
Interoperability is the most important principle of SOA. This can be realized through the use of web services, as one of the key benefits of web services is interoperability, which allows different distributed web services to run on a variety of software platforms and hardware architectures. The Java programming language is already a champion when it comes to platform independence, and consequently the J2EE 1.4 and Java WSDP 1.5 platforms represent the ideal platforms for developing portable and interoperable web services.

Interoperability and portability start with the standard specifications themselves.

Challenges in Moving to SOA
SOA is usually realized through web services. Web services specifications may add to the confusion of how to best utilize SOA to solve business problems. In order for a smooth transition to SOA, managers and developers in organizations should known that:

- SOA is an architectural style that has been around for years. Web services are the preferred way to realize SOA.
- SOA is more than just deploying software. Organizations need to analyze their design techniques and development methodology and partner/customer/supplier relationship.
- Moving to SOA should be done incrementally and this requires a shift in how we compose service-based applications while maximizing existing IT investments.

Conclusion
The advent of web services and SOA offers potential for lower integration costs and greater flexibility. An important aspect of SOA is the separation of the service interface (the what) from its implementation (the how). Such services are consumed by clients that are not concerned with how these services will execute their requests. Web services are the next step in the Web's evolution, since they promise the infrastructure and tools for automation of business-to-business relationships over the Internet.

Interoperability : Interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer.