CLOUD SAAS, SOA AND INTEROPERABILITY
4. CLOUD SAAS, SOA AND INTEROPERABILITY

4.1. Introduction

Service Oriented Architecture (SOA) is called so because of its style in which the enterprise applications are designed and developed as a collection of services, each addressing a specific concern of the business logic. These services (functionalities) are accessible through well-defined interfaces specified for the purpose. Analogous to the concept of services in SOA is the cloud SaaS in Cloud computing. A cloud SaaS refers to a service model wherein the software applications are offered as services to the customers. Extending SOA beyond the enterprise firewalls, into the cloud would help seeking the potential benefits of both SOA and Cloud computing. Since, SOA inherently fosters interoperability it will enhance the integration and interaction among the cloud software services.

This chapter focuses on ‘interoperability’ among cloud services in an SOA-based cloud. Sections 4.2 and 4.3 elaborate the concepts of Web services framework and Service Oriented Architecture (SOA), respectively. Section 4.4 discusses the convergence of SOA and Cloud computing. Section 4.5 discusses interoperability in an SOA-based Cloud SaaS. Section 4.6 illustrates interoperability between two software applications running in the cloud with the help of an example. Section 4.7 draws the conclusion of the chapter.
4.2. Web Service Framework

Web services framework provides the technology platform for realization of SOA. A framework is a collection of one or more architectures, technologies, concepts models and even sub-frameworks. Prior to the discussion on SOA, we are providing an elaborated description of the Web Services framework in this section.

4.2.1. Characteristics of Web Services Framework

The Web services framework is characterized by [54]:

- An abstract, vendor-neutral existence defined by standards organizations and implemented by proprietary technology platforms

- Core building blocks
  - Web services,
  - Service Descriptions
  - Messages

- A communications agreement centered around service descriptions based on WSDL

- A messaging framework comprising of Simple Object Access Protocol (SOAP) technology and concepts

- A service description registration and discovery architecture sometimes realized through Universal Description, Discovery and Integration (UDDI)

- A well-defined architecture that supports messaging patterns and compositions

- A second-generation of Web services extensions

4.2.2. Core Building Blocks of Web Services Framework

The major components at the core of Web Services framework serving as its building blocks are Web Services, service descriptions and messages.
4.2.2.1. Web Services

Web services refer to pure, web-based distributed technology that leverages the concept of standardized communications framework to bridge the enormous disparity that exists between various applications. The services are independent of any operating system or platform technology; and are self-describing and discoverable.

Based on the role a service temporarily assumes during the runtime processing of a message, a service may be classified as:

i. A provider service that implements some functionality and makes it available to other services on the Internet; or

ii. A requestor service that consumes or utilizes an existing service.

Thus, a web service plays the role of a provider when it is invoked via external source and that of a requestor when it invokes another service by sending it a message.

Likewise, based on the nature of logic or functionality that a service provides, a service may be classified as:

i. A business service providing a business-specific functionality;

ii. A utility service providing a generic, non-application specific functionality; or

iii. A controller service that manages and coordinates the tasks of other services.

A service registry is a logically centralized directory of services where the developers can publish new services or find existing ones.

4.2.2.1.1. Web Service Protocol Stack

A protocol stack is the software implementation of a protocol suite in a computer network. A web service protocol stack is a protocol stack that is used to define, locate, implement and make Web services interact with each other.
The four layers of Web service protocol stack, as depicted in Figure 4.1 [61], are:

1. **Service Discovery layer** which centralizes services into a common registry, and provides easy ‘publish/find’ functionality. At present, Universal Description, Discovery, and Integration (UDDI) is being used for this function.

2. **Service Description layer** which describes the public interface to a specific web service and uses the Web Service Description Language (WSDL) for the purpose.

3. **XML Messaging layer** which encodes messages in a standard XML (eXtensible Markup Language) format so that they can be understood at either end. Currently, it includes XML-RPC (XML Remote Procedure Call) and SOAP (Simple Object Access Protocol).

4. **Service Transport layer** which transports messages between the applications and utilizes transport protocols such as Hyper Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and File Transfer Protocol (FTP) etc for the purpose.
4.2.2.2. Service Descriptions (using WSDL)

Service description is the key component that facilitates establishing a consistently loose-coupled form of communication between Web services. Loose-coupling is a condition wherein a service acquires knowledge of another service while still remaining independent of that service; it is achieved through the use of service contracts established by service descriptions. In the Web services framework, WSDL definition is the primary service description document. It describes four critical pieces of data [61]:

1) interface information describing all publicly available functions;
2) data type information for all message requests and responses;
3) binding information about the transport protocol to be used; and
4) address information for locating the specified service.

WSDL service description can be separated into two categories [54][69][70][71]:

4.2.2.2.1. Abstract Description

This description establishes the service interface characteristics without any reference to the technology used to host or enable a Web service to transmit messages. Thus it preserves the integrity of service description regardless of the changes in the underlying technology. The three main elements of the abstract description are – interface, operation and input and output elements or messages. An interface consists of a group of functions, called operations, which a service provides. Each operation represents a specific action performed by the service. An operation, optionally, receives data (input parameter) and sends data (output parameters) represented as input and output messages.
4.2.2.2. Concrete description

The execution of service logic always involves communication between services. The concrete description specifies the connection of abstract description (i.e. interface, operation and messages) to a physical transport protocol to enable this communication.

The main elements of concrete description are – binding, endpoint and service. A binding represents the transport technology that a service uses in order to communicate. The endpoint represents the physical address at which the service can be accessed with a specific protocol. In WSDL, a service is used to refer to a group of related point of contacts for a web service.

Figure 4.2 depicts the abstract and concrete descriptions in a WSDL Document.

Figure 4.2 WSDL Document Displaying Abstract and Concrete Descriptions [54]
4.2.2.3. Messages

Messages are the elements that are exchanged between the services. SOAP is a lightweight, XML-based protocol that facilitates the exchange of information in a decentralized, distributed environment. A SOAP message has three parts – envelope, header and body. The envelope is the root element and serves as a container for the header and the body. The header is optional, and hosts packets of supplementary meta information i.e. information related to the delivery and processing of message contents. The body hosts the actual message content in XML format. The content of a message is also referred to as payload. The message itself may be based on RPC-style or document-style.

4.3. Service Oriented Architecture (SOA)

SOA is emerging as a premier integration and architecture framework for complex and heterogeneous computing environments. It is an architectural style for building software applications that use services available in a network such as the web, thus facilitating the software-as-a-service concept.

Figure 4.3 Services and Interfaces in SOA
The term ‘service-oriented’ represents a distinct approach for separating concerns. The automation logic is decomposed into distinct yet related units of logic, called service, each addressing a specific concern or part of the problem. Each service implements distinct business functionality through well-defined and implementation-agnostic interfaces as depicted in Figure 4.3. The size and scope of the encompassed logic may vary from service to service.

4.3.1. Common Principles of Service Orientation

The key principles governing the services in SOA are [54]:

1. **Loose coupling** – Services maintain a relationship that minimizes dependencies among each other and only requires that they retain an awareness of each other.

2. **Service contract** – Services adhere to a communications agreement, as defined collectively by one or more service descriptions and related documents.

3. **Autonomy** – Services self-control the logic they encapsulate.

4. **Abstraction** – Except for what is described in the service contract, services hide logic from the outside world.

5. **Reusability** – The logic is divided into services with the intent of promoting reuse.

6. **Composability** – Two or more services may be coordinated or assembled to form composite services.

7. **Statelessness** – Services minimize retaining data specific to an activity.

8. **Discoverability** – Services are designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanism.
4.3.2. SOA – Find-Bind-Execute Paradigm

SOA is based on the find-bind-execute paradigm, as depicted in Figure 4.4. The service providers register their service in a public registry so that it is available for use by consumers. A service registry is a collection of services, their description and their addresses. It is used by the consumers to find services that match their requirements. If the registry has such a service that matches the criteria, it provides the consumer with a contract and an endpoint address for that service. The consumer accesses the service, binds to the service by calling its visible interface and invokes the methods provided by the service. The interaction takes place through exchange of SOAP messages and the service is executed [54][72][73]. The terms service provider, service requestor and service registry have been derived from Web services framework which is currently the most suitable technology platform to realize SOA.

![Figure 4.4 SOA's Find-Bind-Execute Paradigm](image)

4.3.3. Key Tangible Benefits of SOA

The adoption of SOA leads to improvements in the construction of automated solutions. The common tangible benefits of SOA are:
1. *Improved Integration and intrinsic interoperability* – The implementation of highly standardized service descriptions and message structures significantly lowers the cost and effort of cross-platform integration for SOA-compliant applications.

2. *Inherent Reuse* – Service orientation promotes designing of inherently reusable services. Though this moderately increases the development efforts but in the long run it appreciably lowers the cost and effort of building service-oriented solutions.

3. *Streamlined Architectures* – As the business solutions are aggregates of services which adhere to the SOA design principles, business solutions characterized by highly optimized and streamlined architectures can be developed with reduced processing overhead and skill-set requirements.

4. *Leveraging the legacy investment* – The wide acceptance of Web services technology has enabled legacy applications to participate in service-oriented integration architectures, potentially lessening the need to replace the legacy systems.

5. *Organizational Agility* – Organizational agility refers to the ability of an organization to accommodate change. An SOA-compliant business solution comprised of loosely-coupled, compose-able, interoperable and potentially reusable services establishes a more adaptive automation environment thereby protecting the applications from the impact of technology evolution.

6. *“Best-of-breed” alternatives* – The vendor-neutral communications framework established by SOA enables developers to choose from the available “best-of-breed” technology, for implementing business solutions.
4.4. Converging SOA and Cloud Computing

Service Orientation has evolved from object-oriented and component-based computing. It enables building agile networks of collaborating business applications distributed within and across organizational boundaries. It utilizes services as the fundamental elements for developing applications or business solutions. These services are autonomous platform-independent computational elements that can be described, published, discovered and accessed over the Internet using standard protocols. In this context, services become the next level of abstraction in the process of creating systems that would enable automation of e-businesses [74]. This way of reorganizing software applications and infrastructure into a set of interacting services is usually referred to as Service Oriented Architecture (SOA). SOA is changing the way the computer software is designed, architected, developed, delivered, consumed and analyzed.

In order to realize the vision of service-orientation where billions of users and services interact in a loosely coupled manner, the resources need to be packaged and offered in an economical, scalable and flexible manner making them more affordable and attractive for IT customers and technology investors. A significant underlying infrastructure that allows such efficient service provisioning is referred to as the Service Cloud. Cloud services are accessed over the Internet via user-friendly web interfaces. They are location agnostic, can be hosted through third party service providers and can be quick to improve based on real-time customer feedback.

Service Oriented Architecture (SOA) focuses on decomposing the business logic into distinct services, which are then orchestrated or choreographed to provide enterprise business solutions. In contrast, Cloud computing paradigm which represents the fifth generation of computing following mainframe computing, personal computing, client-
server computing and web computing [75], focuses primarily on provisioning of dynamically scalable, shared resources as services over high bandwidth linkages. A cloud may host a range of software applications implemented using varied platforms and technologies. But, a cloud should not be looked at as a new architecture. Instead, it should be perceived as another option for storing and running services within SOA [56][74]. SOA may play a significant role in Cloud computing as it would facilitate the development of cloud applications with well-defined interfaces and architectures. An SOA-based cloud would naturally abstract and hide the vendor-specific and platform-specific aspects of various software applications by leveraging the open Web services communications framework and establishing a predictable communications medium for all applications exposed via Web service. Enhancing a cloud with a well-planned architecture of the resources based on SOA will result in a more efficient and effective delivery of cloud services.

Although a cloud may be developed without practicing SOA, a quality ‘Cloud’ requires the individual cloud services to conform to common design principles of SOA in order to fully realize the benefits of reusability, interoperability, federation, and others.

4.5. Cloud SaaS, SOA and Interoperability

Cloud computing architecture is a blend of diverse technologies and platforms. The software applications residing in the cloud must be able to interact with each other transparently, irrespective of their implementation technologies. Interoperability among the cloud SaaS is, therefore, a significant issue in Cloud computing.

Interoperability has been defined in [60] as the ability of two or more systems or components to exchange information and to use the information that has been exchanged.
As mentioned earlier, SOA promotes intrinsic interoperability by virtue of the Web services framework used to realize it. Therefore, building a cloud based on the SOA approach would result in cloud services that are inherently interoperable. This intrinsic interoperability would be achieved by virtue of vendor-neutral communications framework that would enable implementations of highly standardized service descriptions and message structures. Additionally, the loose-coupling among the cloud...
services would result in the independence of service logic. The cloud services would only require being aware of each other and would be able to evolve independent of each other.

As depicted in Figure 4.5, not only the cloud may be SOA-based, but the individual software services deployed in the cloud may also be service-oriented.

4.6. An Illustration of Cloud SaaS Interoperability

As mentioned earlier, a cloud may host a variety of software applications as cloud services which fulfill the business requirements of heterogeneous strata of customers. The cloud services may be implemented on different software and hardware platforms. These services deployed in the cloud may require interacting with each other in order to achieve a common business goal. In this regard, SOA-based cloud architecture would promote inherently interoperable services.

An important aspect of SOA is the separation of the service interface (the what) from its implementation (the how) [72]. The interfaces are defined using XML-compliant WSDL, a core component of the Web services framework and are responsible for exposing the service (functionality) to the client. The service implementation provides the required processing. A client (which may be another service), therefore, need not be concerned with the implementation details of the service in order to use it. Besides, any change in the implementation of the service does not prevent the client from communicating with the service, as long as the interface remains the same.

In order to address the interoperability issue in an SOA-based cloud, we extend further our illustration of OHRS cloud SaaS (described in chapter 3) by including another cloud SaaS – the COHS (Cab_on_Hire System) service. This service may be availed by any cab enterprise. A variety of business functionalities are offered by this cloud software
service such as online bookings and/or cancellations of cabs by the customers, generating reports for the management etc. As mentioned earlier, the OHRS cloud service also offers a range of functionalities to the clients through its interface enabling them to book accommodation, cancel a booking, check availability status, generate reports etc. Next, we consider two hypothetical enterprises – ABCHotels and XYZCabs – that subscribe to these cloud services [67] and provide services to their customers through their respective websites – ABCHotels.com and XYZCabs.com. Also, the two enterprises complement each other in terms of the services they provide, i.e. the ABCHotels hires cabs from XYZCabs for its customers. Similarly, XYZCabs hires rooms for its customers from ABCHotels. This requires a B2B communication between the two enterprises. The interaction is facilitated by the cloud services through simple interfaces – an OHRS_Service interface for OHRS and a COHS_Service interface for COHS. Although, a service may maintain different interfaces for B2B and B2C interactions, for simplicity we are assuming a single interface to enable both.

Let us, for example, consider that ABCHotels wishes to enquire about the total number of their customers who have hired cab(s) from XYZCabs on a particular day. This requires the ABCHotels to invoke a specific method through the COHS_Service interface. As depicted in Figure 4.6, this interaction in an SOA-based cloud is enabled through an XML-based communication framework that uses SOAP messages. The hotel administrator submits a request, from one of the ABCHotel.com web pages, to XYZCabs.com to determine the number of its customers availing the cab service on a particular day. The HTTP request generated is routed via a controlling servlet on the ABCHotel.com web server, which determines that it needs to retrieve the information (raw data) from XYZCabs.com. The servlet obtains this data by using a web service client implemented by ABCHotel.com developers. This client uses the web service
interface published by XYZCabs.com to invoke a method on its server that returns the required information. The method invocation is performed by creating an XML message that contains the method name and any required parameters and then sending it to XYZCabs.com’s server using the SOAP protocol. The value(s) returned by the method call are then wrapped in another XML message and sent back to the ABCHotel.com’s web client. This client extracts the information that it needs and uses a server-side script engine to render it as HTML. The HTML is then returned to the client’s browser. The advantage of using XML instead of HTML is that only raw data is required to be transferred which does not include presentation markups, thereby reducing network traffic. Also, the code required to make a request is much simpler than that required to extract data from an HTML page [76][77].

Figure 4.6 XML-Based Interaction between Cloud Software Services

4.7. CONCLUSION

Cloud computing is a fast emerging computing paradigm where the computations are performed using third-party hardware and software resources hosted in a cloud. In this chapter, we have emphasized on cloud architectures to be based on SOA paradigm. This would ensure improved integration and interoperability among the software services in the cloud. The basic concepts of Web services and SOA have also been elaborated in this
chapter. Web services and SOA offer potential for lower integration costs and greater flexibility of the cloud services. Besides, the interoperability between the software services in an SOA-based cloud has been illustrated with the help of a suitable example. The convergence of SOA and Cloud computing would enable service vendors, service consumers and developers to harvest benefits of both.