A SURVEY OF VARIOUS TECHNOLOGIES FOR SOA ADOPTED BY SOFTWARE INDUSTRIES

Karthik .V, Prof. R.Jaya
P.G. Scholar, Sr. Assistant Professor, Department of Computer Science and Engineering,
New Horizon College of Engineering, Bangalore, Karnataka, India
karthik.v114@gmail.com, jayamanojkumar@gmail.com

ABSTRACT-This paper presents the knowledge about technologies in Service- Oriented Architecture (SOA) and Web services. To integrate the various applications of a large enterprise, we need various standards to remove the heterogeneity problems. The various standards and its responsibilities are depicted in this paper. The standards for discussion are SOAP, WSDL, UDDI, XML, and JAVA. Finally, we depict the technologies and development challenges and directions in the field of SOA and Web services.

Keywords - Service integration, SOAP, UDDI, WSDL, XML.

1. INTRODUCTION TO SOA AND WEB SERVICES

In an SOA, applications are made up of loosely coupled software services, which interact to provide all the functionality needed by the application. Each service is generally designed to be self-contained and stateless to simplify the communication that takes place between them. There are three major roles involved in an SOA:[1]

: Service provider
The service provider creates a service and can publish its interface and access information to a service broker.
A service provider must decide which services to expose and how to expose them. Often, a trade-off exists between security and interoperability; the service provider must make technology decisions based on this trade-off. If the service provider uses a service broker, decisions must be made about how to categorize the service, and the service must be registered with the service broker using agreed-upon protocols.[1].

: Service broker
The service broker, also known as the service registry, is responsible for making the service interface and implementation access information that is available to any potential service requester.

The service broker provides mechanisms for registering and finding services. A particular broker might be public (for example, available on the Internet) or private, only available to a limited audience (for example, on an intranet). The type and format of the information stored by a broker and the access mechanisms used is implementation-dependent.[1].

: Service requester
The service requester, also known as a service client, discovers services and then uses them as part of its operation.

A service requester uses services provided by service providers. Using an agreed-upon protocol, the requester can find the required information about services using a broker (or this information can be obtained in another way). After the service requester has the necessary details of the service, it can bind or connect to the service and invoke operations on it. The binding is usually static, but the possibility of dynamically discovering the service details from a service broker and configuring the client accordingly makes dynamic binding possible.[1]

To summarize, a complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism.[3].

2. BENEFITS OF WEB SERVICES

: Exposing the Existing Function on the Network
A web service is a unit of managed code that can be remotely invoked using HTTP. That is, it can be activated using HTTP requests. Web services allow you to expose the functionality of your existing code over the network. Once it is exposed on the network, other applications can use the functionality of your program.[3].

: Interoperability
Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.[3].

: Standardized Protocol
Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.[3].

: Low Cost Communication
Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web

services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.[3].

3. TECHNOLOGIES

Over the past two years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are:[2].

: Simple Object Access Protocol (SOAP)

SOAP provides a standard packaging structure for transporting XML documents over a variety of standard Internet technologies, including SMTP, HTTP, and FTP. It also defines encoding and binding standards for encoding non-XML RPC invocations in XML for transport. SOAP provides a simple structure for doing RPC: document exchange. By having a standard transport mechanism, heterogeneous clients and servers can suddenly become interoperable. .NET clients can invoke EJBs exposed through SOAP, and Java clients can invoke .NET Components exposed through SOAP. [2].

: Web Service Description Language (WSDL)

WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding. WSDL allows disparate clients to automatically understand how to interact with a web service. [2].

: Universal Description, Discovery, and Integration (UDDI)

UDDI provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points. [2].

4. The Role of WSDL, SOAP, and Java/XML Mapping in SOA

: The Role of WSDL in SOA

WSDL is the interface definition language (IDL) that defines the interactions among SOA components. It provides a standard language for describing how to communicate with a component. Without a standard IDL, you must resort to ad hoc documentation to communicate the interfaces for your SOA components.[4].

Figure 4.1 shows the role of WSDL for SOA Integration as described in this book. The figure provides a UML object diagram depicting the relationship of WSDL in an SOA Integration setting. First, notice the Web Services Platform subsystem where deployment takes place. The top-level class depicted in that subsystem is Service Deployment. Each instance of Service Deployment corresponds to a Web service that is deployed on this platform. Next, notice that Service Deployment contains both an operation (taken from the WSDL interface description) and a Java method. In this manner, you see that a Web service deployment defines a relationship between a WSDL interface description and a Java implementation of that description. More specifically, a Web service deployment defines relationships between individual operations in a WSDL and the Java methods that implement them in that particular deployment.[4].

Figure 4.1 shows that a WSDL interface description contains a types instance (i.e., the wsdl:types element). As you know, this is the top-level element within the wsdl:definitions element that describes the XML Schema types used in the WSDL. Here, you can also see that this particular WSDL‘s types instance contains the schema Orders.xsd that is part of the XML Schema Library in an Enterprise System. It also incorporates the schema Faults.xsd from the Web Services Infrastructure subsystem.[4].

The Enterprise System in Figure 4.1 could be the OMS described in which case, its schema library would include the Orders.xsd schema described there. Other ―infrastructure‖ libraries—like standard schema for fault messages—are also envisioned as part of the SOA framework deployed by an enterprise.

So, as described here, one role of WSDL in the SOA framework is to assemble standard XML types into operations that describe Web services. Another role is as a participant in a Web service‘s deployment.[4].

The preceding section discusses how SOA requires an interface definition language (i.e., WSDL) to describe, in a standard way, how to invoke a Web service component. The WSDL describes an abstract interface (i.e., wsdl:portType and wsdl:operation), and a concrete binding of that interface (i.e., wsdl:binding). As shown in Example 4–2, a WSDL operation is described in terms of abstract messages (i.e., wsdl:message elements). SOAP provides a concrete implementation, or binding for the wsdl:message elements, thereby defining the XML structure of the messages exchanged among SOA components in a standard manner. In this paper, SOAP means SOAP Version 1.1 unless a specific reference is made to SOAP Version 1.2. [4].

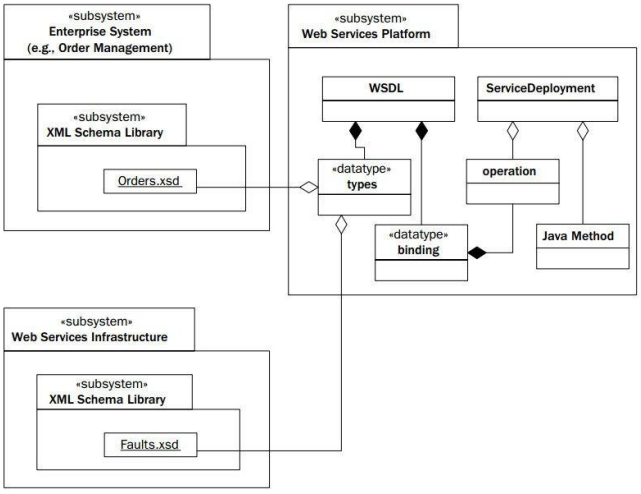

Figure 4.1: The role of WSDL in SOA integration.

: The Role of SOAP in SOA

REST proponents also argue that REST is simpler than SOAP because it does not concern itself with the semantics of SOAP nodes. Along with its envelope structure, SOAP includes a processing model composed of SOAP nodes that transmit and receive SOAP messages, and may relay them to other SOAP nodes. SOAP even goes so far as to prescribe header attributes such as env:mustUnderstand that are used to indicate whether processing of a SOAP header block is mandatory or optional. Complexity creeps in this way because if the ultimate receiver node of a SOAP message cannot process a header block that is marked as env:mustUnderstand, it must reply with a SOAP fault. And the SOAP specification describes a standard envelope structure for SOAP faults. [4].

SOAP node semantics and the associated header processing attributes give the REST advocates most of their ammunition for declaring that SOAP is too complicated. I must admit that I have some sympathy for this point of view. However, at this point, the lack of a standard IDL for REST makes it impossible to work with as a standard for enterprise SOA. [4].

: The Role of Java/XML Mapping in SOA

Java/XML mapping for SOA is accomplished by defining and implementing type mappings. A type mapping is simply a relationship between a Java class and an XML Schema type—for example, corp:AddressType and samples.Address. A type mapping is implemented by a serializer and a deserializer. The serializer converts instances of the Java class into XML instances that conform to the schema. The deserializer does the reverse. [4].

When doing –Start from WSDL and Java‖ development, a large portion of the design process involves defining the type mappings and serializers. For example, suppose you have a Java method such as:

public void updateAddress(String custId, Address addr)

And suppose the corporate standard schema for address is a complex type: corp:AddressType. Then, the WSDL that is deployed to describe the Web service for updateAddress needs to include corp:AddressType as a message part. But furthermore, the serialization subsystem on the platform where the Web service is deployed must be able to access the deserializer for the type mapping . When a SOAP request for the Web service arrives, the deserializer is used to convert the SOAP part to the Java method parameter. This process is illustrated in Figure 4.2. [4].

As discussed in Section 4.3, the dispatching of this SOAP message is based on the wrapper element—custinfo:updateAddress. It gets mapped to the updateAddress method as shown. Below this wrapper element are the two message parts8 —custinfo:custId and custinfo:address. These are mapped to the parameters custId (String) and addr (Address), respectively. This mapping, of the message parts to the method parameters, is not defined in the WSDL. The WSDL contains no information about the underlying Java implementation of the Web service. This property of the WSDL is consistent with the separation of concerns concept discussed in Section 4.2. After all, the consumer of the Web service shouldn‘t have to be concerned with such implementation details. All the consumer needs is the information necessary to construct the SOAP message and send it to the appropriate URL. [4].

So, the type mappings that link the SOAP/WSDL to the Java implementation are not defined in the WSDL, but rather are part of the internal—platform-specific—deployment information associated with the Web service. In the JWS model, these type mappings are defined by the JAXB standard mapping as customized by any annotations.

Figure 4.2: SOAP parts map to Java method parameters.

At this point, we just want to point out that the type mapping process is outside the scope of the WSDL and is a platform-specific issue. Furthermore, you should understand that being able to implement flexible type mappings is a key to the –Start from WSDL and Java‖ development model needed for SOA. As illustrated in Figure 4.2, the key to being able to deploy the updateAddress() method as a Web service with the desired WSDL is to be able to implement the type mapping . [4].

CONCLUSION

In this paper we tried to bridge the technologies of web Services and SOA. Critical Web services infrastructures will be covered, such as WSDL, SOAP, UDDI, Discovery, Composition, Registry, and Web services invocation and relationship binding. How Web and SOA can benefit with each other will also be explored and also about the Role of WSDL, SOAP, and Java/XML Mapping in SOA. Finally, the presenter will depict technologies and development challenges and directions in the field of SOA and Web services. [5].

REFERENCE

[1].http://www.redbooks.ibm.com/redpapers/pdfs/redp4884.pdf
[2].http://gsl.mit.edu/media/programs/south-africa-summer-2015/materials/o'reilly_-_java_web_services.pdf
[3].http://www.tutorialspoint.com/webservices/webservices_tutorial.pdf
[4]. Mark D. Hansen, *SOA Using Java Web Services,* Prentice Hall,2007.
[5].http://www.intersystems.com/assets/SOA_WP-be1cf67974f8d09552742ccdbba0792d.pdf

Karthik.V studying M-Tech (Computer Network Engineering) in New Horizon college of Engineering, marathalli. He completed BE (Information Science Engineering) degree in VTU University in East Point college of Engineering and Technology. His area of interest includes Data Mining, SOA, and Computer Networks.

R.JAYA is working as a Sr.Assistant Professor in the department of CSE at New Horizon college of Engg.She has 13 years of teaching experience. She has done bachelor and master degree in engineering under Anna University. She is doing Ph.D under VTU. Her research interest includes data mining, expert systems,SOA and security.