Assimilation of Web Ministration with Circulated Applications

Veerendra Kumar Sahu1, P. Kiran2, K. Santosh Ch. Rao3
1,2Department of Computer Science & Engineering, Centurion University of Technology and Management, Odisha, INDIA
3Assistant Professor, Department of Computer Science & Engineering, Centurion, University of Technology and Management, Odisha, INDIA

ABSTRACT

A Web service is a distributed application component. Web services distributed computing model allows application-to-application communication. There is nothing fundamentally new about the basic concept and the related technologies. The innovative thing about this is the reach of Web services and its ubiquitous support by literally all major vendors. Most likely, heterogeneity will at the end no longer be an obstruction for distributed applications. Many industrial enterprises acquire disparate systems and applications over the years. The need to integrate these different systems and applications is often prominent for satisfying business requirements and needs. In an effort to help researchers in industrial informatics understand the state-of-the-art of the enterprise application integration, this paper describes the concept of service-oriented architecture (SOA) in conjunction with the Web services technology and covers the core Web services specifications which form a powerful and robust foundation for building distributed systems. It is presented a case study regarding the integration of the Web services with the SAP system for handling interoperability issues. The need to integrate these different systems and applications is often prominent for satisfying business requirements and needs. In an effort to help researchers in industrial informatics understand the state-of-the-art of the enterprise application integration, this paper discusses the architectures and key technologies used for integrating distributed enterprise applications, illustrates their strengths and weaknesses, and identifies research trends and opportunities in this increasingly important area.

Keywords—Web Services, Semantic Web Services, Architecture Approaches, Service revelation, Service Architecture Framework; Service Architecture Performance

I. INTRODUCTION

Because of the level of the application’s integration, the Web services have grown in popularity and are beginning to improve the business processes. In fact, the Web services are being called the next evolution of the Web [1]. Typically large enterprises are supported by hundreds of applications. Many of these applications were written in COBOL on mainframe computers and are referred to as legacy systems. Enterprises critically depend on legacy systems for their day-to-day business operations. It is not unusual for large brokerage firms in the financial services sector to appropriate annual legacy maintenance budget in the order of billions of dollars. Therefore, it is natural for the enterprises to explore ways to reduce legacy maintenance costs. Two primary approaches were pursued. The first approach involves replacing legacy systems with a new application. The latter is designed for better interoperability with other systems, and more importantly easier to maintain and evolve. Enterprise Resource Planning (ERP) systems were introduced to address this need. Though ERP systems were well received stored in different storage servers. Web services provide a promising framework for development, integration, and interoperability of distributed software applications. Wide-scale adoption of the web services technology in critical business applications will depend on the feasibility of building highly dependable services. Web services technology enables interaction of software components across organizational boundaries. In such distributed environment, it is critical to eliminate errors at the design stage, before the services are deployed. Web services provide a promising framework for development, integration, and interoperability of distributed software applications. Wide-scale adoption of the web services technology in critical business applications will depend on the feasibility of building highly dependable services. The remainder of the paper is structured as follows: section 2 provides information about service-oriented architecture (SOA) in conjunction with the Web services technology and the core Web services specifications; section 3 describes a case study on web
services and associated key technologies. The application shows how to integrate Web Services on different platforms and how they allow the interoperability between applications running on these platforms, using the specific web services protocol stack presented in section 2. Section 4 concludes the paper and presents future proposed developments.

A. WEB SERVICES:

The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other’s IT systems behind the firewall.

Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users.

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web services are not tied to any one operating or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications.

Web services do not require the use of browsers or HTML. Web services are sometimes called application services.

Universal Description Discovery and Integration (UDDI)

UDDI contain description of web services and mechanism for requester to get access to the services publication and description. It contains data with combination of white, green and yellow pages. White pages contain information like company name, address, contact information etc. yellow pages contain information like business category, business type etc. while green pages presents that which kind of services the business offer.

Web Service Description Language (WSDL)

WSDL specifies the mechanisms to access a Web service over the network. WSDL files are stored in UDDI as a registry so that a web service requester locates them. It is an XML based language which provides syntactic description of web services.

B. SERVICE-ORIENTED ARCHITECTURE

QUICKLY ASSEMBLE APPLICATIONS AND INCREASE ASSET RE-USE.
Capture service levels and metrics for any protocols or platforms
Pinpoint root causes by using snapshot of the full transaction taken in the event of an error or Service Level Agreement (SLA) violation
Discover services running in your environment and detect any rogue services or consumers

INCREASE AGILITY WHEN INTEGRATING APPLICATIONS

With webMethods, you gain a single integration backbone to connect all IT systems and application silos—from custom, mainframe and legacy apps to ERP, CRM and cloud-based systems. We can help you adopt a well-planned and effective approach to application integration and gain end-to-end application visibility across all systems.

Diagram explaining the approach and monitoring of Application Integration

II. INTEGRATION SYSTEMS

A. Application Integration
Integrate any system across your Digital Enterprise.
Reduce integration costs and development time
Increase business agility
Gain end-to-end visibility across systems

Say goodbye to the high cost and complexity of point-to-point integrations. Welcome a single integration solution that quickly connects any system or application without coding. WebMethods brings together a complete set of integration capabilities, including an analyst-recognized Enterprise Service Bus (ESB), messaging, adapters and monitoring. Used for: enterprise application integration; integrating cloud & on-premise applications; integrating SAP® systems; integrating Oracle® systems; integrating mainframe systems

B. CAPABILITIES
Open, standards-based integration
Our solution—speaks any technology
Integrate virtually any system, including ERP, CRM, cloud-based applications, home-grown applications and legacy systems, such as mainframes
Secure data, encode data and communicate more easily across your Digital Enterprise

Standards-based integration is faster—webMethods supports key standards including HTTP, XML, SOAP and WDSL.

C. Reliable messaging
Guarantee delivery and once-and-only-once delivery of messages
Supports all major messaging patterns, such as publish/subscribe, request/reply and synchronous/asynchronous

D. Multiple document and protocol support
Supports industry-standard protocols, such as HTTP, HTTPS, FTP, FTPS, Java® Message Service (JMS), SMTP, SNMP and file polling
Supports document formats, including XML, flat files, .CSV files and delimited files

III. SERVICE ORIENTED ARCHITECTURE VS. WEB SERVICES

Distributed software systems, and the interactions between components within these systems, can exhibit a high level of complexity and lead to difficulty in the assessment of what system behavior is possible in multiple scenarios.

3.1. SOA
Nowadays, service-oriented architecture (SOA) and web services that enable flexible and loose integration of applications within and across enterprises have become one of the most phenomenal subjects both in academia and in industry. SOA is a software architectural concept that defines the use of services to support the requirements of software users. It is a system for linking resources on demand. In an SOA, resources are made available to other participants in the network as independent services that are accessed in a standardized way. This provides for more flexible loose coupling of resources than in traditional systems architectures. The SOA model treats three main elements that act as a find–bind/invoke-execute. The service provider offers a given service and publishes service description in a service registry. The service requester queries the registry to find a certain service. If found, it retrieves the location of the service and binds to the service endpoint, where the requester can finally invoke the operations of the service. 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
Web services distributed computing model allows application-to-application communication. For example, one purchase-and-ordering application could communicate to an inventory application that specifies the items that need to be reordered or a Web service from a credit bureau which requests the credit history from the loan services, for prospective borrowers. In both cases, the data interaction must be protected to preserve its confidentiality.

Web Services are considered to be the future of the Internet. They are independent of the platform and also of the technology, but in reality they are XML/SML collections of standards which allow the interaction between systems (programs). Heather Kreger, one of the IBM’s lead architects for SOA Standards which developed the standards for Web services, thought that Web Services are like an interface which describes a collection of operations, network accessible throughout the XML standard messages. Web Services have the main role to access different services and different data from different machines, and so they offer to the clients a single public interface.

IV. PROPOSED SYSTEM AND ARCHITECTURE

We are going to integrate web services from various applications across different platforms to design a web application called ACE and also defined routing techniques to provide high availability and scalability. Used for: Banking sectors and Domains. ACE is a front end application where it integrates with various multiple projects to provide effective services to customers. It plays a prominent role as it deals with the clients and customers. The technical focus is on the ACE infrastructure and its connections from clients and to supporting services. Bancorp Customer Service & Monetary Transactions Platform

- Inquiry – provides a single view of the customer including account balances, loan payoff quotes, contact history, customer relationships…
- Customer Maintenance – name & address change, account ownership…
- Service Requests– fee waiver, coupon book order, card disputes, payments errors, statement & check copy…
- Transactions – automated Teller, customer transfers & payments, internal funds transfer mechanism to move money between a customer’s accounts as well as internal operating accounts (DDA & GL)

Users can access some Web services through a peer-to-peer arrangement rather than by going to a central server. Some services can communicate with other services and this exchange of procedures and data is generally enabled by a class of software known as middleware. Services previously possible only with the older standardized service known as Electronic Data
Interchange (EDI) increasingly are likely to become Web services. Besides the standardization and wide availability to users and businesses of the Internet itself, Web services are also increasingly enabled by the use of the Extensible Markup Language (XML) as a means of standardizing data formats and exchanging data. XML is the foundation for the Web Services Description Language (WSDL).

4.1 Semantic Web Service Relevation

The semantic Web service discovery problem consists of locating appropriate services from one or more service registries that are relevant to an input-output request.

V. CASE STUDY REGARDING THE INTEGRATION OF WEB SERVICES ON DIFFERENT PLATFORMS

This section describes a case study on web services and associated key technologies, offering practical tests in order to support the previous presented. The application shows how to integrate Web Services on different platforms and how they allow the interoperability between applications running on these platforms, using the specific web services protocol stack presented in previous sections.

The problem we have modeled is as follows: "A company that sells used hardware in the fields of retail and food industry wants to integrate some mobile terminals, which are going to be used in order to scan the bar codes of the store’s hardware equipments with an ERP system, in our case SAP." Using a mobile terminal, we want to make different types of storage-specific operations, such as: reception, delivery and material’s inventory.

The solution we present for integrating the mobile devices with SAP - consists in developing a web service on a .NET platform, that can be used as a proxy between the two platforms.

Brief description of the scheme

Integrated with SAP Business Object Repository (BOR) is an object-oriented repository that contains SAP objects and SAP interfaces, and also their components, such as methods, attributes and events. In SAP Web Application Server, SOAP runtime provides us the mechanism of using SOAP protocol to call and access the RFC functions (Remote Function Call) via HTTP. A web service in SAP can be seen as an RFC function. The interoperability between the Microsoft .NET platform and the SAP system is done with the help of a component, the SAP Connector for Microsoft .NET. It supports SAP Remote Function Call (RFC) and Web Services and it allows the development of various applications on the .NET platform. The Proxy class generation, which enables the calling for web service (RFC function) is based on WSDL. In this case, it can be said that SAP connector acts as a proxy between SAP and the .NET platform. In order to integrate the mobile terminal with SAP, it has been developed a web service that becomes the wrapper class for the SAP connector and also it becomes a proxy between the application from the terminal and SAP. In order to integrate the Web Service with both distributed systems some steps must be taken:

Stage 1: Developing and publishing the Web Services

Using the developing platform Visual Studio.NET, a series of appropriate methods for the data exchange is constructed. In this initial step the .Net connector is integrated to SAP. If the SAP authentication was successfully done, the description of the available RFC functions it is brought from SAP with the help of UDDI and WSDL. Since WSDL is based on the XML standard the description of the functions is brought into a file with a .sapwsdl extension in XML format. Based on the WSDL and on the .NET connector, the proxy class, that contains the signature of the functions from the file, is being generated in order to become available for calling by other work methods. In the example from below is shown a description of a RFC function from SAP on a proxy call basis, function that allows the over taking of the materials from the system:

Stage 2: Rendering the Web Service

At this stage, after the construction of the web service, it will be rendered in the application from the mobile terminal which is developed on the .Net platform. If the service is available, similar to Stage 1, the description of the service in the application is brought with the help of UDDI and WSDL.

Stage 3: Calling methods of service

After rendering the contract and after the generation of the proxy class to call the service, in the mobile device application one can call the service methods. Exchanging data between applications in the background, in the form of SOAP messages is done asynchronously and at runtime. If there is a large volume of data, due to the interoperability based on XML standard,
the process may be slowed because of the need for parsing of the XML messages in different types of objects. This exchange of messages is done by following these steps:

**Step 1: Request the mobile application service**

While calling a method from the proxy class, on a SOAP framework basis, a request is developed, in XML format. This message is attached to the SOAP - and will on a HTTP protocol basis to the service. If the service is not available and if there are no different restrictions on the network, next will be followed step 2, otherwise the application ends.

**Step 2: Request service from SAP**

The request was received by the service and it will be forwarded to SAP through connector and the proxy class. Based on the SOAP Runtime protocol, which provides access and the calling to the RFC functions, all the messages are sent in XML format via HTTP, and they are parsed as objects and BOR specific structures.

**Step 3: The SAP’s answer to the service**

The result of the request from Step 2 is sent in XML format by the SOAP Runtime and it is parsed by the .NET connector into the service’s specific objects.

**Step 4: The answer of the service to the mobile application**

The result from Step 3, as a response from service to customer, it is sent like in Step 1. The XML message is transformed by SOAP’s framework into specific objects from the development environment.

VI. CONCLUSIONS

Web services play a similar role with older technologies such as Remote Procedure Call (RPC), Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), but also offers several advantages over these. The great advantage of Web Services is that they can integrate different platforms and allow the interoperability between different distributed systems. The Integrity and the interoperability can be addressed with the SOA’s help in a two-stage process that involves publishing and orchestrating the web services. The Independence with respect to various platforms is due to common standard that they all have as a basis, the XML. Using Web services provides to the developers the opportunity to create high quality applications more quickly. In this paper, the authors came up with a pattern of using and integrating the Web services, proofing the interoperability between multiple distributed applications running on different platforms.

REFERENCES

[1] Understanding Web Services


