Woa Based Implementation of SOA

[1]Ashish Verma, [2]Ruchi Dave, [3] Pooja Parnami
[1,2,3]Dept. of CSE, Suresh Gyan Vihar University, Jaipur, Rajasthan, India

Abstract
Service-Oriented Architectures (SOA) is an Emerging approach that addresses the requirements of loosely coupled, standards-based, and protocol independent distributed computing. A Distributed Computing is always required a tight coupled relationship between all working services. Basically SOA provides a large number of objects that are working in modular services as reusable software components. Generally there are no any alternative for SOA to provide flexibility and reduction in the cost of services which are basically used in the IT field as reusable components. This functionality is provided by the Enterprise Service Bus (ESB) that is an integration platform that utilizes Web services standards to support a wide variety of communications patterns over multiple transport protocols and deliver value-added capabilities for SOA applications. But in this Context we are introducing the "WEB 2.0" which is used to provide reusable IT components dynamically.
In this paper we will introduce the methodology of design WOA using the concept of SOA. The big picture will follow the existing SOA model. In particular, this WOA methodology comprises conceptual as well as realization issues and breaks WOA design down into three distinct phases.

Keywords
Design Methodology, Reusable Components and Web oriented Architecture

I. Introduction

The World Wide Web Consortium (W3C) for example refers to SOA as *'A set of components which can be invoked, and whose interface descriptions can be published and discovered'*. We see similar definitions being used elsewhere; it's a very technical perspective in which architecture is considered a technical implementation. This is odd, because the term architecture is more generally used to describe a style or set of practices—for example the style in which something is designed and constructed, for example Georgian buildings.
Service-Oriented Architectures (SOAs) provide the basis of distributed application frameworks (W3C 2004b) where software components are provided as modular and reusable services. The benefits of a SOA are seen in the flexibility of business processes which consist of loosely coupled services, and the resulting potential cost decrease, complexity reduction, reusability potential, and high flexibility. Conceptual modeling is an important factor here, as it not only refers to data modeling but also needs to take process design into consideration. It's would be easy to conclude that the move to Service Orientation really commenced with Web services—about some years ago. However, Web services were merely a step along a much longer road. The notion of a service is an integral part of component thinking, and it is clear that distributed architectures were early attempts to implement service-oriented architecture. What's important to recognize is that Web services are part of the wider picture that is SOA. The Web service is the programmatic interface to a capability that is in conformance with WSnn protocols. So Web services provide us with certain architectural characteristics and benefits—specifically platform independence, loose coupling, self-description, and discovery—and they can enable a formal separation between the provider and consumer because of the formality of the interface. Moreover, the various standards for easing" the creation of a SOA (e.g., Web services, UDDI, SOAP, WSDL).

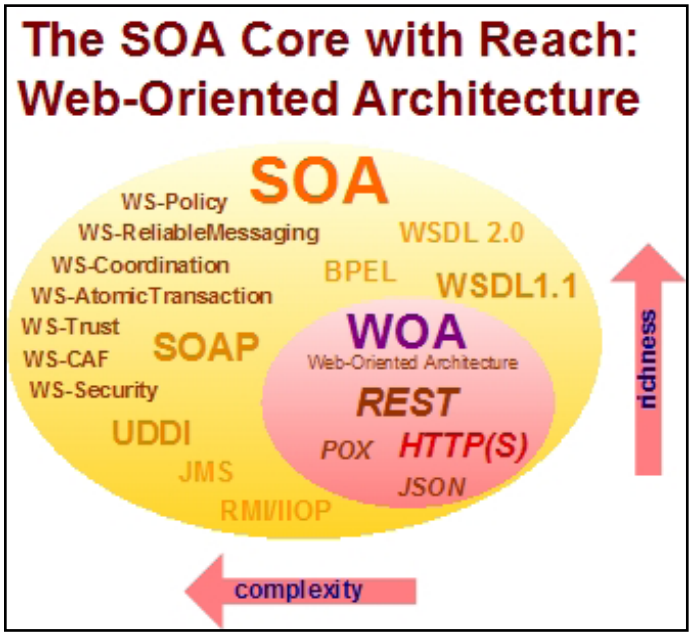

Fig. 1: The Summarized Map of SOA and WOA

On the other hand, recent developments in the context of what is commonly termed "Web 2.0" show how easy it can be to link or compose ("mesh") IT components dynamically, so that original SOA goals like flexibility, reusability, or reduction of complexity can indeed be reached by relatively simple means. Examples include mashups based on Google Maps (like www.housingmaps.com) or applications like Yahoo!Pipes (pipes.yahoo.com) these are based on Web Application Programming Interfaces (Web APIs), which allow using the functionality of a Web application by a simple (most commonly REST-based) service layer. An interesting and emerging concept in this context is the Web-oriented architecture (WOA), which represents a specialization of a SOA obtained by emphasizing the use of simple Web 2.0 technologies and standards. Its important aspect is the fact that no additional standards have been defined, but existing ones such as HTTP, SSL,
Or XML are employed. As far as REST and WOA are concerned, you don't need anything more complex than HTTP, one of the most scalable, proven, and widespread protocols on the planet, along with HTTP's verbs: GET, POST, PUT, DELETE.

II. WOA Versus SOA

There are some basic differences in terms of design and practical considerations in between WOA and SOA, both are used to develop reusable IT components but still they have some competition issues and just an evolution of focus which are as follows:

- The only real difference between traditional SOA and the concept of WOA is that WOA advocates REST, an increasingly popular, powerful, and simple method of leveraging HTTP.
- SOAP, the most common Web service standard also has mandatory dependency system that can be problematic, but using REST it removes this problem.

- WOA itself is a reflection of the Classic Reach vs. Rich Argument, in that richness is an outcome of robustness and complexity but cuts down how much reach you have to others, particularly in heterogeneous communities. The modern SOA technologies and architectures have indeed become a near-morass of complex standards, protocols, and products.
- REST doesn't handle enterprise issues like two-phase commit, messaging, asynchronicity. Whereas the SOAP which used in SOA will handle all of these issues.
- Although both are using for the purpose of Web services with different standards but we can say WOA is the most interoperable, easiest to implement, and most highly scalable technique for building great, open Web services that anyone can use.

III. Requirements and Goals

The existing methodologies for designing a SOA mostly focus on Web Services as the core connection element. In contrast, a WOA uses Web services as well as REST-ful services, i.e., services that can be invoked via REST. As a consequence, a WOA approach is not bound to a specific technology. While Web services can be described by a WSDL document for REST-ful services there is no explicit standard for their description. However, a reasonable solution for the time being is the Web Application Description Language (WADL) which is much simpler than WSDL, yet allows the description of important features of a REST-ful service such as resource, input, output, and even a textual description. Most services today come with textual descriptions only (as can be seen, for example, at www.programmableweb.com), so no matching algorithm can be specified. Therefore, most services have to be "discovered" by the software architects themselves. This can be done in different ways, but our approach currently assumes that the search for suitable services is done by hand in the second phase. As long as no accepted semantic service description is available and used for a large portion of the services offered on the Web. One of the important points of our approach is the usage of well-known and well-understood formal languages in every phase, including BPMN for process descriptions and WSDL as well as WADL for the functional description of individual services. The advantage of BPMN is that it is easy to read and still allows for a powerful visual design of processes.

The following simplified steps are identified to model, construct, and implement a WOA:

- Development a set of inter-connected business process models.
- Refinement of these process models with functional details of each service task.
- Provisioning of the process models with explicit data flow (especially a specification of which data is used as input for a service request).
- Implementation of the defined process models (in a development environment or workflow engine).

IV. Design and Implementation Methodology

In our Example we are taking Vehicle Booking System which primarily includes Check Vehicle Module, Booking of Vehicle Module and Payment of Vehicle Module, from the generic requirements we can easily identified the details requirements and we have chosen Flow Diagrams to show the flow of processes and business modules which are using in our "Macro-Booking" System as per shown in the figs. 2, 3, 4

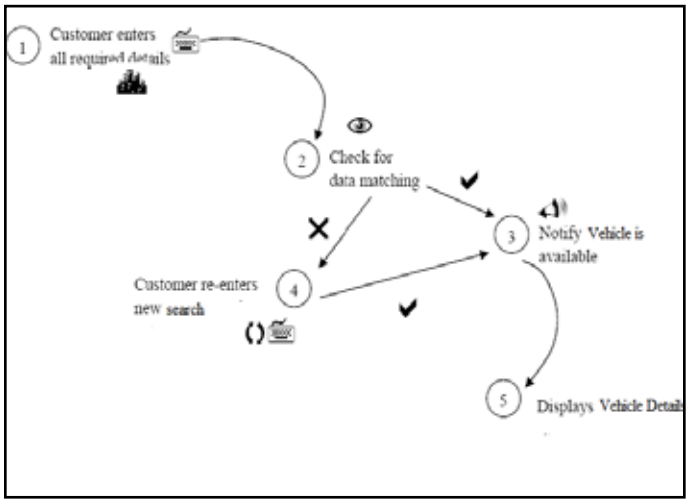

Fig. 2: Flow Diagram of Check Vehicle Module

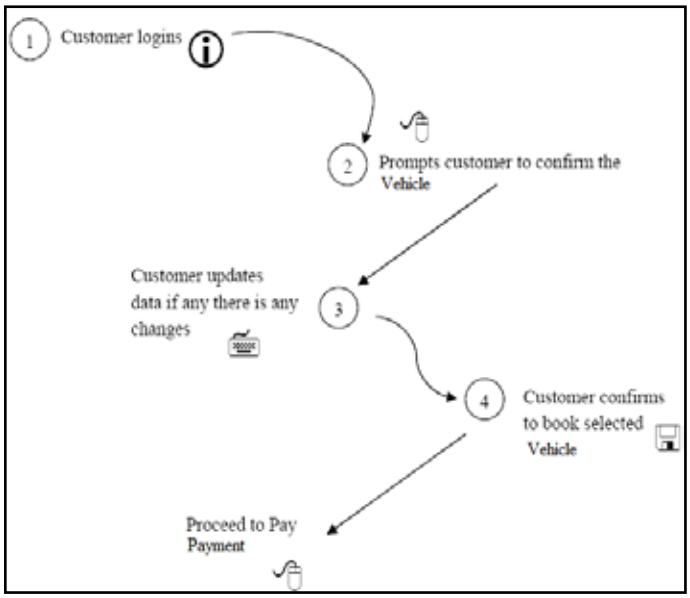

Fig. 3: Flow Diagram of Booking of Vehicle Module

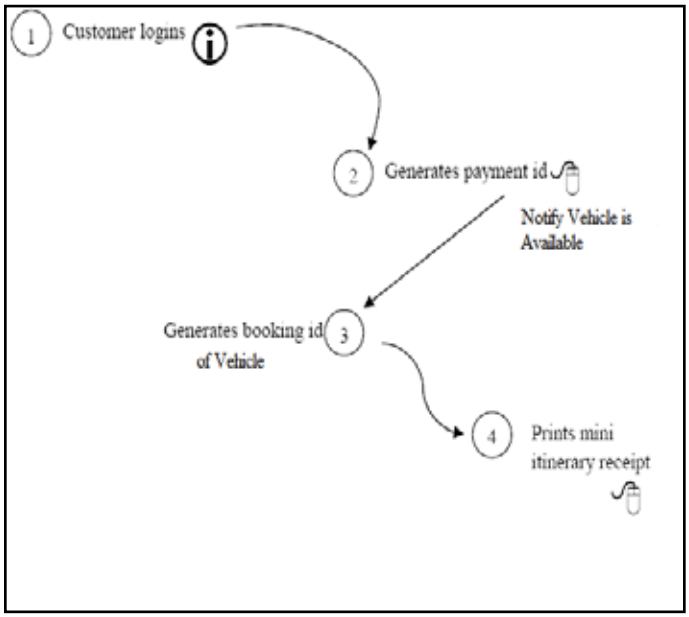

Fig. 4: Flow Diagram of Payment of Vehicle Module

The Business Scenario of (Check Vehicle Module) is as Follows in fig. 2:

- Firstly Customer will enter in the Service by entering its search details like Vehicle type, Vehicle Module and Color etc. [1].
- If its requirements does match with our system then service will notify that particular vehicle is available [2-3].
- After that search is available then it will show the details of particular vehicle [5]
- If customer requirements does not match then customer have to re-enter again with new search and checking details of vehicle [4].

The business scenario of (Booking of Vehicle Module) is as follows in fig. 3:

- Now if customer wants to book a vehicle then first they have to login [1] after that service will prompt the user to confirm its order or booking [2].
- Customer can check its details of vehicle and can also be made some changes if he wants [3].
- After all of these steps Customer will confirm to book a particular vehicle.
- Now the next module comes is Payment in which customer have to enter for further processing.

The business scenario of (Payment of Vehicle Module) is as follows in fig. 4:

- Again customer has to login first [1].
- This login will generate a UNIQUE PAYMENT ID for a particular customer [2].
- After confirmation of payment service will generate a BOOKING ID of a particular vehicle [3].
- In this way we can print a mini itinerary receipt of vehicle [4].

Table 1: Web Service Operations

Web Method Name	Web Service Description
Check Availability of Vehicle	This Web service will check the availability of vehicle as per user entered criteria.
Confirmation of Vehicle	This web service will make confirmation of vehicles.
Booking of Vehicle	This web service is used to book the vehicle.
Payment of Vehicle	This web service is used to make payment of vehicle.
Check Credit Validity	This Web service is used to check the validity of declined credit.

V. Implementation Results

As we have mentioned earlier that we want to create a REST-ful Web Service, for this purpose we have taken example of Vehicle Booking System, the Representational State Transform (REST) is not a standard but it uses the standards like:

- HTTP
- URL's
- XML/HTML/GIF/JPEG/etc (Resource Representations)
- text/xml, text/html, image/gif, image/jpeg, etc (MIME Types)

In Principles of designing REST-ful Service we should adopt some key points these are as follows:

- The key to creating Web Services in a REST network (i.e., the Web) is to identify all of the conceptual entities that you wish to expose as services. See figure 2, 3, 4.
- Create a URL to each resource. The resources should be nouns, not verbs. For example, do not use this:

http://www.parts-depot.com/parts/getPart?id=00345

Note the verb, getPart. Instead, use a noun:

http://www.partsdepot.com/parts/00345

- All Resources should be accessible via HTTP GET, PUT, POST and DELETE Methods.
- Categorize your resources according to whether clients can just receive a representation of the resource, or whether clients can modify (add to) the resource.

For further implementation we have used VISUAL STUDIO .NET, SQL, HTTP Methods(GET,PUT,POST and DELETE) and some generic development tools for creating interface apart from interface we have used Microsoft IIS Web Server for responding to the Clients. HTML, AJAX, CSS and JAVASCRIPT is used to integrate the Client Interface.

VI. Conclusion

In this paper we have presented a WOA design methodology which abstracts from technology and complex standards and only uses simple Web standards like HTTP, SSL, and XML for communication and simple data flow syntax to describe data mappings for any request within a process.

As a result of implementation, WOA approach cut-off development time of the web services. The services created are reusable and flexible to be integrated with other web service applications. Conclusively, WOA is a buzzword today and many organizations and industries are in race to adopt WOA in order to have competitive advantages for services delivery. However, it is important to ensure that the right approach is selected and the right capabilities are provisioned to ensure successful realization. It is recommended to select an approach or methodology based on the primary business drivers for adoption practice.

As mentioned in the Introduction, we have seen that our methodology is not just purely conceptual, but rather a hybrid one that meshes conceptual as well as physical aspects of a WOA. We consider this a consequence of the fact that a WOA no longer needs to follow the strict layering of a SOA, but we believe that it is exactly this aspect what will make them more successful than SOAs.

References

[1] W3C (2003, May 14). Web Services Architecture, [Online] Available: http://www.w3.org/TR/2003/WD-ws-arch-20030514/

[2] [Online] Available: https://www.ibm.com/developerworks/library/ws-soa-design1

[3] W3C (2004b),"Web service architecture", [Online] Available: http://www.w3.org/TR/ws-arch/wsa.pdf

[4] "Web services business process execution language version 2.0".

[5] W3C (2007),"Web services description language2.0", [Online] Available: http://www.w3.org/TR/wsdl20-primer/

Ashish Verma Received his Bachelor of Engineering Degree from University of Rajasthan in Year 2009 in Information Technology Branch and he is Doing his Master in Technology from Suresh Gyan Vihar University in Software Engineering, He is a Former Software IT Trainer of HCL,India and Worked as an Assistant Professor in IT Department at Poornima Group of Colleges, right Now he is serving to his best knowledge in ANAND INTERNATIONAL COLLEGE OF ENGINEERING, Jaipur. He Presented and Published 6 National and 2 International Paper in Different Conferences and in Journals, His area of Interest is Software Engineering and Web Services.