



Service Oriented Architecture

WHITE PAPER

Author: Shrikantha Prabhu

Enterprises which embark changing trends vouch with service oriented architectures as they go well with the strategic changes. Service oriented architectures make better investment sense also. After the dotcom bubble burst, enterprises had to take cost-cutting measures. Web services were touted to be an easy option, as they were considered easy on integration.

Barring a few exceptions, Web services are still being seen as cost minimizers. However, this white paper takes a divergent view and tries to advise why the service oriented architecture, the architecture on which the Web services are built, makes strategic sense. The paper then takes the stock on the current state on this new approach of service oriented architecture. Also, it suggests some common pitfalls, the companies should avoid, going the SOA way. The paper concludes with the current challenges while managing service oriented architecture.

Table of Contents

INTRODUCTION	3
DRIVERS FOR SERVICE ORIENTED ARCHITECTURE	3
SERVICE ORIENTED ARCHITECTURE-101	6
SOA- PATTERNS AND CHALLENGES	9
CONCLUSION	11
ABOUT THE AUTHOR	11
ABOUT WIPRO TECHNOLOGIES	12
WIPRO IN SOA	12

Introduction

After the dotcom burst, the companies worldwide went on a cost cutting spree. Web services were presented as easy on integration and hence could be used to reduce the total cost of operation. Any one, using SOAP, considered himself/herself to be one of Web service implementers and a candidate for drumming up lower integration costs. Overselling the lower integration costs via Web service route lowered the strategic importance of adopting what is now known as service oriented architecture. Barring a few exceptions, notably an HBR case study¹, Web services are still being seen as cost minimizers. This white paper however takes a divergent view.

Drivers for Service Oriented Architecture

The adoption of the service-oriented architecture is imperative because of emergence of following three drivers. The drivers are acceleration in strategic changes, advent of grid computing and overall rise in complexity of business.

Acceleration in Strategic Changes

It is seen that the strategic horizons are shrinking. Earlier the strategies were considered to be lasting for at least a decade if not more. These days the disruptive technologies have ensured the constant state of turmoil where the organizations are forced to change their strategies midway. A change in the strategy of the organization percolates into change in the IT operations and CIO looking for systems and architectures which are change friendly. Worldwide, the M & A activities have increased and every M & A throws up integration challenges for the CIO. He has to spend his energies in getting two disparate systems work.

Advent of Grid Computing

Grid computing is going to become the largest inflection point the IT Industry has witnessed so far. Big hardware providers such as IBM envisage first commercial usage of grid computing to start as early as 2006.

Grid computing and allied technologies will make computing power available on tap just like another utility. You do not own hardware, but you buy a computing service from a provider who owns the farm of hardware. You will use his hardware and pay for the usage. Or computing infrastructure will become a service available on wires.

This is where a lot of interesting things are going to happen. Hardware costs will disappear from the capital budget of the company. Eventually the software and packages too follow the path of the hardware. The software will be hosted on the computing service provider's servers and true pay per usage concept gets a boost. Big ERP vendors are already seeing this written on the wall and moving to pay per usage model thus envisaging that future of packages lie on service model.

¹ John Hagel III and John Seely Brown - Harvard Business Review, October 2001

When grid computing becomes pervasive, the software will cease to exist, as a product but becomes a service. You do not buy licenses for the lifetime, but rather pay per use.

In such a scenario, IT departments:

- Will have to integrate with a package available on a grid
- Or, will also have to make their own applications available on the grid

Once the applications move to grid following there will be two major demands on the architecture of the applications

- Applications will have to become client agnostic

It will be no longer possible to predict the kind of client who is accessing the application. The client may be a day trader accessing the stockbroker’s website using a WAP phone or Internet browser, or can also be another intelligent application that is trying to access the services using a SOAP call.

- Applications should allow rapid scaling up

The application performance will be put to test because of frequent and rapid scale ups that can happen.

Overall Rise in Complexity of Business

It is a given fact that the complexity in business is rising. The current architectures seem to falter in case of very complex situations. Everyone who is aware of software estimation swears by the rule “double the functions means triple the cost”. This happens because rise in functionality increases the integration costs and that is the pitfall of the tightly coupled applications.

The rise in integration costs can be shown by the following illustration (Figure 1).

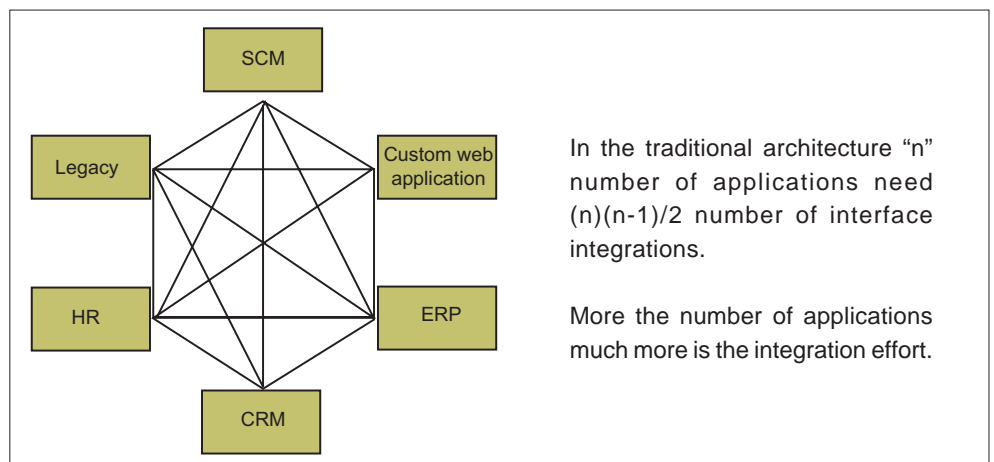


Figure 1: The rise in integration costs

In an enterprise, if n applications are used, for seamless operation of the enterprise, $(n)(n-1)/2$ interface integrations are necessary. For a company with large number of applications, integration costs can become major part of the operations budget.

Increase in this type of complexity calls for a newer approach in architecture, namely Enterprise Service Bus. Here,

- The applications should be build to integrate.

The application should expose its functionality in a standardized manner and publish its capabilities to a common registry.

- And, incrementally the applications should be deployed on a logical bus.

This bus, called as enterprise service bus will allow, standards based, service oriented backbone capable of connecting hundreds, or even thousands of application endpoints.

Figure 2 shows the approach.

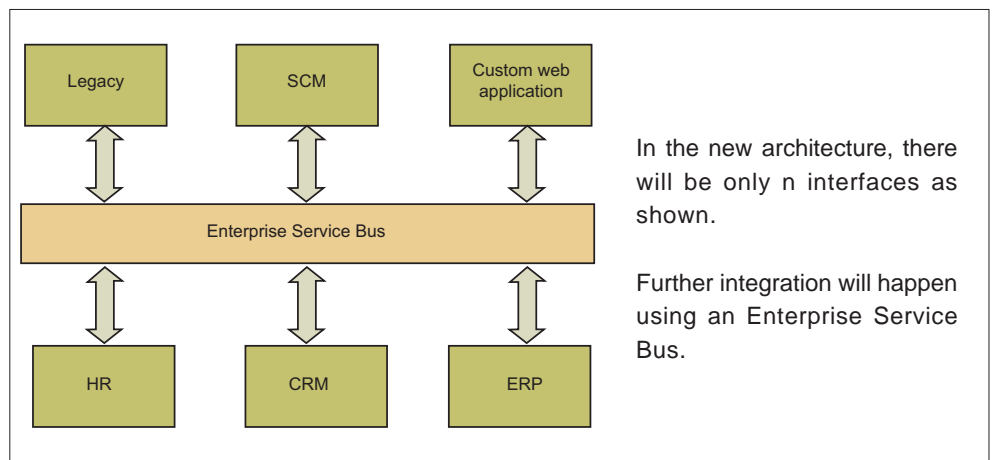


Figure 2: Enterprise Service Bus approach

If an application has to effectively integrate with Enterprise Service Bus, application should be built with SOA.

Service oriented architecture allows easy integration because:

- Expensive integration tools are not necessary
- Easy expansion of the system is possible
- Low recurring cost

To begin with, Enterprise Service Bus may be built using traditional EAI middleware, however as the technology matures we can expect all the infrastructure services to be available from the application servers or SOAP toolkits.

Summary of Strategic Drivers

Thus we see that new architecture has following requirements.

- It should allow easy integration- in light of rising strategic changes
- Server should become client agnostic – in light of advent of grid computing
- Application should allow rapid scaling up – in light of advent of grid computing
- Application should allow curtailing the spiraling complexity – in light of rising complexity

Service oriented architecture can allow realization of the above requirements.

Service Oriented Architecture-101

What is Service Oriented Architecture

To understand what is service oriented architecture we need to first define what a service is and how it is different from the concept of object as envisaged by OOP.

A service is a logical entity, an organization exposes to network. A service is suited for consumption by machines rather than humans. A service may use physical resources like databases programs, devices or even humans. Service is stateless i.e., it does not remember a service request after it was serviced. In addition, services are generally called from across the network. Since network calls for considerable overhead, due to latency, the service should provide significant functionality to the caller in a single call. Or, the service calls should be chunky.

The above observation that the services are stateless and also that service calls should be chunky are two major differences between objects as envisaged by OOP and services as envisaged by SOA.

Another difference between objects and services is in the way the integration happens. In OOP, the integration happens using the implementation of interfaces. If client wanted a callback, the client had to implement callback interface. Anyone implementing a callback interface could be integrated into a framework. This integration was tight, in the sense, it made changes in the server, which affect the client. In the case of services the integration occurs, using exchange of messages at runtime. This integration is loose.

Service oriented architecture is defined as an architecture that divides the problem space into set of services keeping in mind principles such as chunkiness, statelessness and helps in creating framework of applications that can integrate by passing messages. It is not necessary to use Web service related technologies such as WSDL, UDDI, and SOAP while creating SOA architecture. You can realize SOA even without using WSDL, UDDI, and SOAP. However the services we design should be exposed to network and if that needs to be done we need to use the Web service standards such as WSDL, UDDI, and SOAP.

We have already pointed out that chunkiness and statelessness are two features of the SOA. Another feature of service oriented architecture is the ability of service oriented architecture to support multiple client interfaces.

Initially, the SOAP toolkits unwittingly allowed, and encouraged object view of the business domain by making SOAP RPC. However, increasingly document approach is being favored over RPC. Still transition from object-oriented architecture to service oriented architecture is not straightforward. The architects who are used to object oriented architecture have to think away from objects, but start thinking in terms of services.

Good SOA Practices

The following are some of the best practices in SOA.

Decide on the semantics first

Before embarking on a SOA journey, identifying and tabulating the semantics of the business is a must. The provisioning of the service over network can throw up interesting possibilities of co-operation between other services. You need to be able to present to any other service or service provider the semantics of your business in clear terms if you want such a co-operation to surface. Further, thinking in terms of semantics makes you focus on messages that need to be transferred between two services rather than their implementation.

Reduce Chattiness

As already described, the service should be more coarse-grained than RPC. A service will have network overheads. Making two services chatty will not be a very good performance decision.

We may not be able make all the services as coarse-grained services. In such case we need to use a layered approach.

There will be a layer of fine-grained services which are transparent to the user. The user accesses the services through a coarse-grained service which aggregates the functionality of several participating fine-grained services as shown in Figure 3.

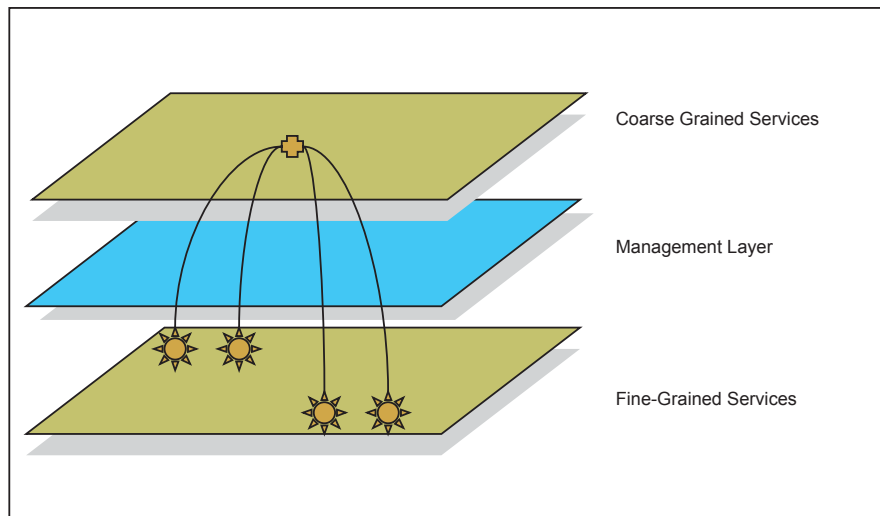


Figure 3: Layered approach - aggregating the functionality of several fine-grained services

The fine-grained services may be managed by a Web service management tool such as AmberPoint, Confluence or Wisiba. Aggregation of several fine-grained services into a coarse-grained service may be achieved using integration products such as WebLogic Integrator, webMethods etc.

Prefer being stateless

The services should strive to be stateless. The session management should be preferably left to the client. There are also mechanisms such as WS-Coordination or WS-Context to allow embedding of the context in the messages itself.

Work with interoperability in mind

In the SOA world you cannot assume a single vendor world. Your service has to interact with other services probably implemented by some other vendor. It is not enough if you ascertained the WSDL compatibility between .net and J2EE (two prominent camps in web service).

You also need to ascertain many other things such as reliable messaging, security, transaction support and so on. For example, as far as reliable messaging is concerned there is further split in the camps with some vendors following their own mechanisms for reliable messaging.

Design service with “sharing” in mind

The services by nature are shared. They are to be shared across different application clients as well as different users. The value of the service is more, if more people can use it. Hence a service should be designed with sharing in mind. Sharing also means the service should be designed for some specific business objective and also be generic enough to be used across multiple scenarios. It may be difficult to achieve objectives of specificity as well as generality in the intended business use for a service. In such a case we may follow principle of dividing the services in question into fine-grained and coarse-grained services as already discussed.

Interface design is the essence of SOA

Services are meant to be shared programmatically. Interface defines the rules for such programmatic exchange. Interface should list all input data and all response data. Interface should carry additional information such as exceptions that can help such programmatic sharing.

A service is as good as its interface clarity. The design of the service implementation is secondary in the design of SOA, the design of the service interface is primary²

Loose Coupling is good SOA

One principle to ascertain good SOA implementation is ascertaining loose coupling during design time. Loose coupling means having no affinity to a particular consumer. The service producer should not make any assumptions on the purpose or nature of the service consumer.

Service should deliver clear business value

SOA is as much about consumption of services as about provision of service. SOA architect should also invest time in checking if the service that is intended to be developed is already available elsewhere and if yes, should make a build or buy decision.

² Introduction to service oriented architecture, Y Natis, R Schulte, Strategic Planning Note from Gartner

The organization should stick to providing services, which are based on their key differentiators. For example an inventory management service for a consumer good manufacturer may use another service to get current exchange rates. The exchange rate service may be provided by a financial institution. It is not a good business practice to scatter your resources on some thing, you are not good at. Businesses should provide services built around their differentiators and their competencies. For everything else, they can be consumers of services offered by best of the breed service providers in that area.

SOA- Patterns and Challenges

This section lists out few of the design patterns that can be used to ease the SOA implementation.

This also discusses the management of services and challenges therein.

Design Patterns

Adapter

This pattern is used to provide a different interface to an existing software component to make it more compatible to what the client is expecting. In the service-oriented architecture, the clients may even be human beings accessing the service over browsers. In such a case an adapter can be designed using portlets to provide a human UI to the service.

Facade

Facade pattern is used to reduce the coupling between client and server. This can be used to achieve right kind of granularity between a client and a service. Using this pattern, the performance of the service may be improved because network calls may be reduced.

Proxy

This pattern is used to provide placeholder for another object. This design pattern may be used in testing the Web service using technique of mock objects (<http://www.mockobjects.com>).

Controller

This pattern is used to separate the data from the presentation. If you already have a framework implemented and want to 'Web service enable' your operations supported by the framework, choose this pattern. You can create a new view of Web service retaining the model and controller elements with minimal changes.

1 Introduction to service oriented architecture, Y Natis, R Schulte, Strategic Planning Note from Gartner

Managing the Services

As we have discussed, the services are likely to be hosted on the grid. Or the services will not have any idea of the infrastructure on which they are running. In earlier situations where the applications were hosted on a real hardware, one could implement the management of the applications at the operating system level.

In the new scenario of virtualized hardware, the management of the service should be given a thought from the early stages itself. Management cannot be plugged into the service later. From the ground up, the service should be designed to manage itself on the virtual hardware. These Managed Services can use CIM or SNMP. CIM allows correlation of various metrics and allows easy fault detection. The Java applications can use JMX.

Managed services should be built with intelligence to maintain and manage certain service quality levels. Currently there are following challenges while implementing managed service oriented applications

- **Monitoring should be for health of application and not just of its components**
When we have loosely coupled applications, the required functionality is realized by interaction of various loosely coupled components. From the service view point, monitoring only the component's health would not suffice. We will have to correlate the health status of various components and deduce the overall health of the service.
- **Troubleshooting problems without knowing where to look**
Troubleshooting the Managed service applications will be very tricky due to distributed nature of applications. Hence, adequate logging should be ensured.
- **Billing and provisioning**
This is the toughest part in managing a service oriented application. Since the service is made stateless, it has no idea of who is accessing and using the service. Yet the billing details have to be generated. When a service in turn uses some other service which is external, appropriate charges are to be paid to the external service. Charges may not be flat but may be dependent on the quality of the service. Tracking the usage and quality levels across various parties that make up the service is a challenge in itself.
- **Orchestration**
The services need not be built from ground up. They may be realized by composing a few of other existing services to provide a consolidated functionality. In such a scenario, the services need to be orchestrated.
- **Versioning and configuration management**
Online nature of service envisages service availability to be nearly hundred percent. Further services being loosely coupled the integration happens using declarative documents such as WSDL. If a service implementation changes and a new end point is added to WSDL making all the clients aware of the changes to the service endpoint is a challenge. Further since the services themselves depend on other services where an SLA may exist, we may need to retire a service as and when the SLA on related services expires. The versioning problem can be addressed using tModels in UDDI.

Conclusion

Service oriented architectures are necessary as they go well with the strategic changes sweeping across and hence they make better investment sense. There are subtle differences between service oriented architecture and object oriented architecture. Notably the services tend to be chunky, stateless and can support multiple clients.

Some design patterns can be used to implement service oriented architecture quickly. Managing the service oriented application can be challenging because of distributed nature. Billing and provisioning the SOA can be tricky unless services are built as managed services with adequate thought over future management needs in the virtual hardware world.

About the Author

Shrikantha Prabhu has been working with Wipro as an Architect in the Web Service Practice. He holds a master's degree in Information Systems Management, from Indian Institute of Technology, Kharagpur. In his career spanning nine years, he has worked with a fortune 500 metal maker, a startup, a big five consulting firm and a fortune 5 financial powerhouse in diverse roles such as software engineer, consultant and team leader and a product architect. His other interests apart from architecture are network management and mathematics.



About Wipro Technologies

Wipro is the first PCMM Level 5 and SEI CMMi Level 5 certified IT services company globally. Wipro provides comprehensive IT solutions and services (including systems integration, IS outsourcing, package implementation, software application development and maintenance) and research & development services (hardware and software design, development and implementation) to corporations globally.

Wipro's unique value proposition is further delivered through our pioneering Offshore Outsourcing Model and stringent quality processes of SEI and Six Sigma.

Wipro in SOA

Wipro has been implementing Web service and system integration projects with a loyal list of extremely satisfied customers. Wipro has been ranked as the best software vendor in the world, for Web service and system integration in terms of its ability of execution. Wipro is also ranked very high in terms of completeness of the vision for Web service, much higher than some of the big five consulting firms (Gartner's magic quadrant on Web services) and system integration.

For further information visit us at: <http://www.wipro.com/soa>

For more whitepapers logon to: <http://www.wipro.com/insights>

© Copyright 2004. Wipro Technologies. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without express written permission from Wipro Technologies. Specifications subject to change without notice. All other trademarks mentioned herein are the property of their respective owners. Specifications subject to change without notice.

Worldwide HQ

Wipro Technologies,
Sarjapur Road,
Bangalore-560 035,
India.
Tel: +91-80-2844 0011.

U.S.A.

Wipro Technologies
1300 Crittenden Lane,
Mountain View, CA 94043.
Tel: (650) 316 3555.

U.K.

Wipro Technologies
137 Euston Road,
London, NW1 2 AA.
Tel: +44 (20) 7387 0606.

France

Wipro Technologies
91 Rue Du Faubourg,
Saint Honoré, 75008 Paris.
Tel: + 33 (01) 4017 0809.

Germany

Wipro Technologies
Horn Campus,
Kaistrasses 101
24114 Kiel.
Tel: +49 (431) 77 55 713.

Japan

Wipro Technologies
911A, Landmark Tower,
2-1-1 Minatomirai 2-chome,
Nishi-ku, Yokohama 220 8109.
Tel: +81 (04) 5650 3950.

The Netherlands

Wipro Technologies,
Polaris Avenue 140,
2132 JX Hoofddorp,
Tel: +31 (23) 5544 630.

www.wipro.com
eMail: info@wipro.com

Wipro Technologies

Innovative Solutions, Quality Leadership