Service Oriented Architecture: Potential Benefits and Challenges

ZAIGHAM MAHMOOD
School of Computing
University of Derby
Kedleston Road, Derby, DE22 1GB
UK

Abstract: - Globalisation, tighter economies, business process outsourcing, ever increasing regulatory environments and knowledgeable consumers are forcing the large enterprises to transform the way they provide their business and services. Businesses are required to be agile and flexible and IT managers are being asked to deliver improved functionality while leveraging existing IT investment. In this climate, Service Oriented Architecture (SOA) is proving to be an attractive approach to Enterprise Application Integration and other solutions that they seek. SOA promises better alignment of IT with business, effective reuse, interoperability and reduced costs of development. However, like any approach, it has its limitations and therefore posses a number of challenges. In this paper, we introduce the SOA approach, present the benefits it offers and discuss the inherent issues and challenges. The objective is to provide enough background information that enterprises wishing to embark on the road to SOA have a better understanding of this approach.

Key-Words: - Service oriented architecture, SOA, Enterprise application integration, EAI, WEB services, Service-orientation.

1. Introduction

Service Oriented Architecture (SOA) is an emerging architectural style for developing and integrating enterprise applications. It is an organisational and technical framework to enable an enterprise to deliver self-describing and platform independent business functionality [1] providing a way of sharing functions, typically, business functions, in a widespread and flexible way. Knorr and Rist [2] define SOA as a broad, standalone and standards based framework in which services are built, deployed, managed and orchestrated in pursuit of an agile and resilient IT infrastructure. British Computer Society's definition suggests that *SOA is about the evolution of business processes, applications and services from today's legacy-ridden and silo-oriented systems to a world of federated businesses, accommodating rapid response to change, utilizing vast degrees of business automation* [3]. This architecture aims to provide enterprise business solutions that can extend or change *on demand* as well as provide a mechanism for interfacing existing legacy applications regardless of their platform or language. It is being seen as a new approach to enterprise application integration, which provides a closer alignment between business and IT systems.

In the rest of this paper, we first establish the need for SOA and briefly outline the SOA framework and technologies. Then, in sections 4 and 5, we mention the potential benefits that SOA aims to achieve and discuss the limitations and inherent issues. In the last section, we present summary and conclusions.

2. The Need for SOA

Enterprises have invested heavily in large-scale applications software such as ERP (enterprise resource planning), SCM (supply chain management), CRM (customer relationship management) and other such systems to run their businesses. The infrastructure is often heterogeneous across a number of platforms, operating systems and languages. There is often a huge duplication of functionality and services resulting in a waste of valuable resources and poor response times. Increasingly, the business and IT managers are being asked to deliver improved functionality of services while leveraging existing IT investment as well as provide the following:

- respond to business changes with agility
- meet the demands of ever changing business environment, customers and partner organisations

- provide continuous business process improvement
- support new channels of business
- provide better customer support
- feature an architecture that supports *organic* business [4].

One solution is to develop architectures that allow easy integration of the existing and new enterprise applications. However, the integration technology solutions are often proprietary which present issues of inoperability because of vendor lock-ins, tight coupling, complexity of services and issues of connectivity [5].

The Web Services (WS) technology and SOA offer better opportunities for enterprise application integration with the added benefits of reduced costs, easier maintenance, greater flexibility and improved scalability. SOA, with its loosely coupled nature, allows enterprises to plug in new services or upgrade existing services [4] and provide opportunities to increase business agility and be able to respond *on demand*. It allows enterprises and their IT systems to be more agile to the changes in the business and environment.

3. SOA Elements and Technologies

In a SOA, the business and technical processes are implemented as services. Each service represents a certain functionality that maps explicitly to a step in a business process [6]. In this context, a service is a software component that can be reused by another software component or accessed via a standard-based interface over the network. An important aspect of service-orientation is the separation of service *interface* (the WHAT) from its implementation or *content* (the HOW). The interface provides service identification, whereas, the content provides business logic.

Zimmermann [7] suggests three levels of abstractions within SOA:

- Operations: units of functions operating on received data, having specific interfaces and returning structured responses
- Services: logical groupings of operations
- Business processes: actions or activities to perform specific business goals by invoking multiple services.

In terms of service-orientation, we can envisage three types of services [8]:

- Infrastructure services: to include security, management and monitoring
- Business-neutral services: to include service brokers and notification, scheduling and workflow services
- Business services: to include services based on the business domain e.g. credit card validation, address verification and inventory checks.

SOA uses a *find-bind-execute* paradigm. The main components include:

- Service providers – components (available to *consumers*) that execute business functions using given inputs and producing outputs
- Service consumers – components that use services published by service providers
- Service registry – a repository containing service descriptions for service consumers to know how services may be accessed.

Service Providers build services and offer them via an intranet or Internet. They *register* services with service brokers and *publish* them in distributed registries. Each service has an interface, known as *contract* and functionality, which is kept separate from the interface. The *Service Consumers* search for services (based on some criteria) - when found, a dynamic *binding* is performed. In this case, the service provides the consumer with the *contract* details and an *endpoint* address. The consumer then *invokes* the service.

Services, implemented as Web Services (WS) are delivered using technologies such as eXtensible Markup Language (XML), Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description Discovery and Integration (UDDI).

Many proprietary SOA tools and frameworks have also been produced for the development of web services and implementation of SOA. Majority of these are difficult to use and do not deliver the business benefits claimed. They lack vital capabilities like configuration control or testing prior to deployment. Hohpe [5, 8] believes the next generation tools will provide facilities for testing and debugging as well as provide support for monitoring and management. For a review of a number of such products from vendors such as BEA, Eclipse, IBM and CapeClear, refer to Mahmood [17].

4. SOA Promise

SOA provides an opportunity to achieve broad-scale interoperability while offering flexibility to adapt to changing technologies and business requirements. If implemented correctly, SOA offers the following benefits [9, 10]:

- Loosely coupled applications and location transparency
- Application connectivity and interoperability
- Alignment of IT around the needs of the business
- Enhanced reuse of existing assets and applications
- Process-centric architecture
- Parallel and independent development
- Better scalability and graceful evolutionary changes
- Reduced costs of application development and integration
- Easier maintenance
- Reduced vendor lock-ins.

The SOA represents a new way of developing systems; it promotes a shift from writing software to assembling and integrating services. Underlying platform implementation becomes irrelevant as standard interfaces and message exchange patterns provide integration, both within and across enterprises. However, to support the goal of SOA, the infrastructure must support flexibility, heterogeneity, distributed development and management [9].

5. SOA Limitations and Issues

SOA is a much better architecture as opposed to distributed client server architecture and it can bring huge benefits in the form of code reuse, better integration and improved responsiveness to business needs. However, the eternal battle between flexibility and efficiency exists in just the same way as it always has. SOA also requires a large upfront investment by way of technology and development. It may cost a great deal and the Return on Investment (ROI) could take a long time to materialise. Overall [12] mentions the following downsides to SOA:

- Since any service can invoke any other, each service needs to validate *completely* every input parameter. This will have implications by way of response time and overall machine load.
- It is possible that a bug or corruption introduced in a well-used service takes out the entire system, not just a single application.

Managing services metadata is another obvious challenge. As services keep exchanging messages to perform tasks, number of these messages can go into millions even for a single application [13].

Although, Web Services provide a sensible implementation platform, many infrastructure services (e.g. security, systems management, interface contracts) are not yet fully defined. Finding a service that is at the right level of abstraction is also a challenge.

Issues, inherent due to the very nature of service-orientation, can be summarised as follows:

- Coarse granularity: This may mean that 1) testing and validating every combination of every condition in a complex service may well become humanly impossible; 2) one service trying to serve a dozen masters may lead to spaghetti code and therefore introduce massive inefficiency and 3) a generic service, because of its coarse granularity, cannot be easily optimised for efficiency [12].
- Loose coupling: It is an architect's dream but making a system distributed adds a new level of complexity and therefore as Fowler [16] puts it, it can become a *developer's nightmare*.
- Integration of services: This can be a complex task especially when there is a lack of skilled people to work in a SOA based environment [13].
- Service interoperability: When web services are used to exchange SOAP messages over HTTP, encapsulating XML data, integration of services in heterogeneous environment can become a serious issue.
- Evolutionary development: Building and updating services is fine. However, if applications continually require additional functionality, and these requests are granted, the entire system may become unstable [12].

Other challenges and limitations can be summarised as follows:

- WS standards: These standards are open and amorphous. Many are still working drafts.

Higher-level services and security have not been standardised at all. This could result in rework of existing code to conform to new and evolving standards [8].

- Internet protocols: They are not designed for reliability, guaranteed delivery or order of delivery so the service or the consumer needs to ensure that messages has been delivered or received in a timely manner [18].
- Interoperability: Common implementations of SOA use web services to exchange messages over the Internet. These encapsulate XML data. Problems may be encountered when integrating these services in heterogeneous environments.
- Development tools: Vendors are producing tools to enable organisations to reap the benefits of SOA but a majority of these are early releases and based on evolving standards. This may also result in rework of existing code once the standards are agreed upon [8].
- Network connections: If the architecture and services are highly interconnected then even a partial network failure may create a huge problem for the relevant organisations. Currently, although the required technology is available, there is no guarantee that infrastructures will be robust with enough redundancy to cope with the system downtimes.
- Security: When using open standards, a service is much more open to other services and applications than a monolithic application [13] and thus security becomes an issue. As mentioned before, Internet protocols also lack reliability. Although, WS-Security addresses such issues, there is a considerable amount of work that still needs to be done.
- Application ownership: SOA blurs the boundaries of application ownership so who owns what becomes an issue.
- Training: It takes time to acquire knowledge of new technologies. There are too many technologies and at the present moment, not enough expertise available in the market. For an organisation adopting this architecture, thorough understanding of the underlying technologies remains highly critical [8].
- Governance: UK firm Gartner warns that *SOA projects will fail unless they are tightly managed and audited* [14]. Since SOA is a new paradigm, governance issues are not properly understood.

Moving to SOA is not a straight forward transition. It requires a shift in how we compose service-based applications whole maximising existing IT investment [15]. SOA requires building systems at a business level, not just at the IT level. Delivery of services needs to be focused on the business requirements. Once the business processes and architectural structures have been defined, one can think about the technology needed to deliver a fully operational SOA. The development should be incremental.

6. Conclusion

Adopting SOA can be difficult for business and IT executives. It requires enterprises to identify the services infrastructure to deliver the required business solutions. Although SOA promises huge gains as it is based on sound principles of coarse-grained, loosely coupled, standards-based, interoperable, reusable services, there are also numerous challenges such as change in mind set, huge initial investment, unreliability of Internet protocols, evolving standards and the newness of the approach.

The bandwagon for SOA is large and many companies are already jumping on it. Enterprises need to be planning for it and at least be ready. They need to be aware of the vendor hype and be extra vigilant when committing huge sums of money in a technology that is still evolving.

In this paper, we have discussed the characteristics and the potential benefits that SOA promises as well as the relevant technologies. Limitations and inherent issues have also been discussed in detail. The objective is to provide some useful background information for enterprises wishing to embark on the road to SOA.

References:

[1] Cartwright I and Doernenburg E, Time to jump on the SOA bandwagon, *IT Now, British Computer Society*, May 2006

[2] Knorr E and Rist O, 10 Steps to SOA, *Info World - San Mateo*, Vol. 27, Issue 45, pp 23-35, Nov 2005

[3] Barnes D, The service oriented architecture: more than just another TLA?, *British Computer Society.*
Retrieved Feb 2007, from
www.bcs.org/server.php?show= ConWebDoc.3041

[4] Kodali R R, What is service oriented architecture? *JavaWorld.com*, 13 June 2005.
Retrieved Feb 2007, from http://www.javaworld .com/javaworld/jw-06-2005/jw-0613-soa.html

[5] Hohpe G, Developing Software in a service-oriented world, *Whitepaper, ThoughtWorks Inc.*, Jan 2005.

[6] Groves D, Successfully planning for SOA, *BEA Systems Worldwide*, 11 Sept 2005

[7] Zimmermann O, Krogdahl P, Gee C, Elements of service-oriented analysis and design, *IBM Corporation*, 2 June 2004

[8] Hohpe G, Stairway to Heaven, *Software Development*, May 2002

[9] Sonic Software Solutions, Service oriented architecture.
Retrieved Feb 2007, from
http://www.sonicsoftware.com/solutions/service_oriented_architecture/index.ssp

[10] Johnson, B, The benefits of service oriented architecture, *Objectsharp Consulting.*
Retrieved March 2007, from
http://objectsharp.com/cs/blogs/bruce/pages/235.aspx

[11] Clark L, SOA gathers pace in the enterprise, *Computer Weekly, UK*, 26 Sept 2006

[12] Overall D, Have we been there before?, *Opinions, Computer Weekly, UK*, 11 April 2006

[13] Wikipedia, Service-oriented architecture. Retrieved 28 Feb 2007, from
http://66.102.9.104/ search?q=cache:nQKzo1LexEsJ :www.sics.se/~olgace/Dictionary.doc+wikipedia+%27service-oriented+architecture%27&
hl=en&ct=clnk&cd=5& gl=uk&client=firefox-a

[14] Saran C, SOA will fail without governance: warns Gartner, *Computer Weekly, UK*, 12 Sept 2006

[15] Mahmoud Q H, Service-oriented architecture and web services: the road to enterprise application integration, *Sun Microsystems Inc.*, April 2005

[16] Fowler M, Patterns of enterprise application architecture, *Addison Wesley*, 2002

[17] Mahmood Z, Service oriented architecture: tools an technologies, *Proc WSEAS Int Conference, Crete, Greece*, July 2007

[18] Colan M, Service-oriented architecture expands the vision of web services – part1, *IBM Corporation*, 21 April 2004.