Service Oriented Architecture: A New Paradigm for Enterprise Application Integration

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

Abstract: - Service Oriented Architecture (SOA) is an emerging architectural style for developing and integrating enterprise applications. Businesses are required to be agile and flexible and IT managers are being asked to deliver improved functionality while leveraging existing IT investment. Globalisation, tighter economies, business process outsourcing and ever increasing regulatory environments are forcing the large enterprises to transform the way they provide their business and services. SOA promises better alignment of IT with business, seamless integration of business functions and reduced costs of development and, therefore, enterprises are moving towards this new paradigm. However, there are numerous obstacles that must be overcome before the *promise* can become a reality. In this paper, we introduce the SOA approach, present the benefits and challenges it offers and provide guidance for building and implementing SOA. 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 applications integration, EIA, Web services, Service-orientation, XML.

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 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 (EAI), which provides a closer alignment between a business and its IT systems.

In the rest of this paper, we first establish the need for SOA and mention the potential benefits that SOA aims to achieve. Then, in sections 4 and 5, we outline the SOA framework and technologies and discuss the limitations and inherent issues. Section 6, presents suggestions for building and implementing the SOA paradigm and the last section, presents 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:

- Business agility
- Meeting customer and partner organisations' demands
- Continuous process improvement
- New channels of business
- Business architecture that is *organic* in nature [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 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
- Seamless connectivity of applications and interoperability
- Alignment of IT with business needs
- Enhanced reuse of existing assets and applications
- Process-centric architecture
- Parallel and independent development
- Better scalability, ease of maintenance and graceful evolutionary changes
- Reduced costs of application development and integration
- Reduced vendor lock-ins.

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

In this view, business processes consist of a number of operations, executed in accordance with certain business rules, to achieve certain objectives. A service is an exposed piece of functionality. According to Erl [19], services need to be governed by the following basic and core principles:

- Services are autonomous and self-contained
- Services share a formal interface, called *contract*, which is platform independent
- Services are loosely coupled
- Services abstract underlying logic – underlying logic is invisible to outside world
- Services are composable, allowing logic to be represented at different levels of granularity
- Services are reusable and stateless.

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.

In term of an enterprise information architectural framework, considering a business from the viewpoints of enterprise, computational and technology, SOA can be further divided into a number of architectural models each representing a different logical layer as follows [18]:

- Business architectural model: this refers to the system as a combination of higher level coarse grained services that provide some business value
- Application architectural model: this refers to the system that exhibits realisation of services in the business architectural model as a combination of smaller much finer gained services
- Implementation architectural model: this refers to the system realised in a certain manner using certain software and hardware systems.

SOA uses a *publish-find-bind-execute* paradigm as shown in Figure 1. 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.

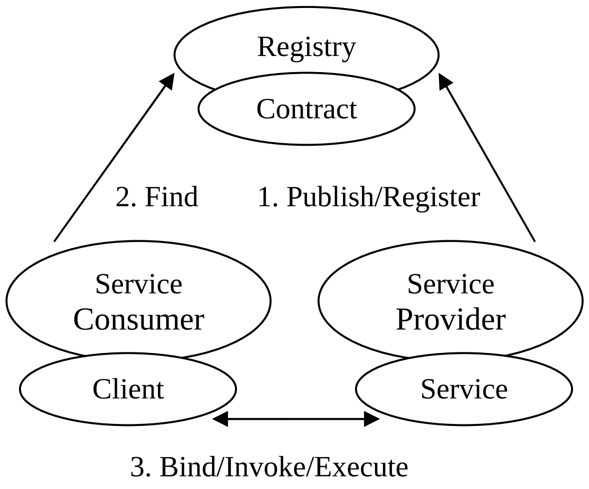

Figure 1: *Publish-Find-Bind-Execute* paradigm

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) and, 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).

5. SOA Limitations and Issues

SOA can bring huge benefits in the form of code reuse, better integration of existing enterprise applications as well as new applications and improved responsiveness to business needs. However, SOA also requires a large upfront investment by way of technology and development as well as a different mindset and availability of required expertise. Overall [12] mentions the following downsides to SOA:

- Since services can invoke other services, each service needs to validate *completely* every input parameter – this has implications by way of response time and overall machine performance.
- A bug or corruption introduced in a well-used service can propagate and take out the entire system.

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 service may well become impossible; 2) one service trying to serve a dozen masters may lead to spaghetti code and introduce inefficiency and 3) a generic service cannot be easily optimised for efficiency [12].
- Loose coupling: Although an architect's dream it can become a *developer's nightmare* [16].
- Integration of services: This is an issue especially when there is a lack of skilled personnel to work in a SOA based environment [13].
- Service interoperability: When web services exchange SOAP messages over HTTP, encapsulating XML data and integration of

services in heterogeneous environment is not easy.
- Evolutionary development: When 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: Many standards are still working drafts. This could result in rework of existing code to conform to new and standards when agreed [8].
- Internet protocols: They are not designed for reliability, guaranteed delivery or order of delivery so the consumer needs to ensure that the message has been delivered or received in a timely manner [17].
- Development tools: Vendors are producing tools but a majority of these are early releases This may also result in rework of existing code when the standards are agreed upon [8].
- Security: Internet protocols, as they currently exist, lack reliability. Although, WS-Security addresses such issues, there is a considerable amount of work that still needs to be done.
- Training: There are too many relevant technologies and it takes time to learn and use new technologies.
- Governance: UK firm Gartner warns that *SOA projects will fail unless they are tightly managed and audited* [14].

Although, Web Services provide a sensible implementation platform, many infrastructure services (eg 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.

6. SOA Implementation

Implementing SOA is not an easy task. It requires a shift in how we compose service-based applications whole maximising existing IT investment [15]. 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].

SOA requires building systems at a business level, not just at the IT level. Delivery of service 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.

For a successful transition to a SOA, one can view the SOA life cycle stages as being the following:
- Development: of loosely coupled and reusable application components - exposed as services and used by business processes and other applications.
- Discovery: by organising a service directory, to act as ‘yellow pages’ based on an open standard e.g. UDDI.
- Integration: of services with applications and other services, including data transmission services, reliable message delivery mechanism etc.
- Orchestration: of services to ‘sequence’ the required services to fulfil a particular business task using acceptable standards e.g. Business Process Execution Language (BCPL).
- Deployment: of integrated and orchestrated services for the 24-hour-7-day availability.
- Monitoring: of processes in real time and analysis and resolution of issues and difficulties as well as analysis of key performance indicators, application of business rules and other metrics.
- Management: of the entire processes of development and execution and analysis with respect to process improvement.

As for the environment, infrastructure and controls, there is a need for the establishment of the following three key elements for successful transition to a SOA:
- Governance framework: consisting of rules and policies to ensure that risks are managed, duplication is avoided, processes are managed and discoverable, standards are followed and changes to the system are appropriately controlled.
- Process change mechanism: to align business processes to IT services in a more explicit manner so that information and data are created, updated and managed more

efficiently – so that processes can adopt more easily to the changing environment.

- Business process management: to make Services more visible in business processes and manage business processes in real time, avoiding errors and duplication – and to construct or *orchestrate* new processes from existing services.

In simple terms, the organisations can use the following strategy [5]:

- Start with the main business processes that span multiple business units.
- Identify services to support these business processes.
- Identify operations from existing systems that could expose as services to support these business processes.
- Identify common supporting infrastructure services.
- Review the process undertaken and incrementally expand by including more business processes (and services).
- Build an application catalogue for future reuse to reduce cost of development.

7. Conclusion

SOA 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 requirement of a change in mind set, huge initial investment, unreliability of Internet protocols, evolving standards and the newness of the approach. Adopting SOA for EAI is therefore far from straight forward.

The bandwagon for SOA is large and many companies are already jumping on it. SOA is an effective paradigm for EAI and therefore enterprises need to be planning for. They also 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, the potential benefits that SOA promises as well as the inherent issues and limitations. Relevant framework, associated technologies and guidance on building and implementing a SOA has also been discussed. 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
http://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] Colan M, Service-oriented architecture expands the vision of web services – part1, *IBM Corporation*, 21 April 2004.

[18] Stojanovic Z, Dahanayake A, Sol H, Agile modelling and design of SOA.
Retrieved March 2007, from
www.cs.ucl.ac.uk/staff/g.piccinelli/
eooows/documents/paper-stojanovic.pdf

[19] Erl T, Core principles for service-oriented architectures, August 2005.
Retrieved Feb 2007, from
www.Looselycoupled.com/opinion/2005/erl-core-infr0815.html