The Promise and Limitations of Service Oriented Architecture

Zaigham Mahmood

***Abstract*—**In today's markets, business enterprises are required to deliver improved functionality and provide on-demand services, while leveraging existing IT infrastructure and investment. They are expected to be agile and dynamic. It is the globalization, tighter economies, business process outsourcing and ever increasing regulatory environments that are forcing businesses to transform the way they provide their business and services. In this context, Service Oriented Architecture (SOA) is proving to be an attractive approach that promises better alignment of IT with business vision, more effective reuse, better interoperability, reduced costs of development and more efficient operation of business applications. However, like any other approach, it has its limitations and inherent issues. This paper introduces the SOA paradigm, presents the benefits it offers and discusses the inherent limitations and challenges. The objective is to provide enough background information so that enterprises, wishing to embark on the road to SOA, have a better understanding of this approach.

***Keywords*—**Service oriented architecture, SOA, Enterprise application integration, WEB Services.

I. Introduction

Service Oriented Architecture (SOA) is an emerging architectural style for developing and integrating enterprise applications. It is an organizational and technical framework to enable an enterprise to deliver self-describing and platform independent business functionality [1] providing a way of sharing business functions and services 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, language or mode of operation. It is being seen as a new approach - a silver bullet - to enterprise application integration to provide a closer alignment between business vision and IT infrastructure and systems.

In this paper, we first establish the need for SOA and briefly outline the SOA framework and technologies. Then, in sections IV and V, 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.

II. 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. These applications are usually stand-alone and the infrastructure is often heterogeneous across a number of platforms, operating systems and languages and modes of operations. Because of the self-contained nature of these applications, 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, ensuring availability as and when required, while leveraging existing IT investment. They are increasingly being expected to provide the following:

- Respond to business changes and customer requirements with agility,
- Meet demands of ever changing business environment and partner organizations,
- Provide continuous business process improvement,
- Support new channels of business including provision on the Internet,
- Provide better customer support and ensure their satisfaction,
- 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. The Web Services (WS) technology and SOA appears to be an answer – that provides opportunities for better business applications development and integration with the added benefits of reduced costs, easier maintenance, greater flexibility and improved scalability.

Manuscript received December 23, 2006; Revised April 6, 2007.

Z. Mahmood is with the School of Computing, University of Derby, DE22 1GB, UK (phone: 00-44-1332-591733; e-mail: z.mahmood@derby.ac.uk).

III. SOA Elements and Technologies

In a SOA, the business and technical processes are implemented as services. Each service represents a particular 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 the following two components:

- The service *interface* (the WHAT) – which provides service identification, definition of parameters and conventions for transferring the service results back to the consumer.
- The service *content* or implementation (the HOW) – which provides business logic as stateless computation.

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 or combinations of other services.
- Business processes: actions or activities to perform specific business goals by invoking multiple services.

Services can be divided into three main groups [8]:

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

At the highest level of abstraction, SOA uses a *find-bind-execute* paradigm, as shown in Fig. 1. The main elements include:

- Service providers – *components* available to consumers that execute business functions using given inputs and producing outputs.
- Service consumers – components or *customers* that use services published by service providers.
- Service registry – *repository* of service descriptions so that consumers know where services exist and how services may be accessed or used.

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.

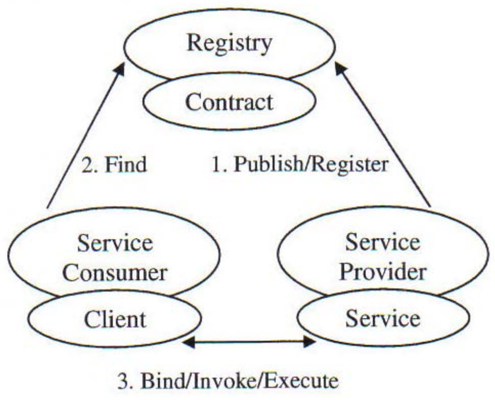

Fig. 1 Publish-Find-Bind-Execute paradigm

Services, usually 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).

Technologies such as XML, SOAP, UDDI and WSDL address the basics of interoperable services and ensure that clients can find and use the required services irrespective of where the clients reside or what technologies are used to create or use the services. XML is fundamental to WSs; WSDL is used to describe WS interfaces; SOAP provides a protocol for transferring messages; and UDDI acts as a repository of available WSs. However, for the SOA to become a mainstream IT practice, other standards such as those to do with security and management of services need to be added. Such standards, referred to as WS-*, are already emerging and organisations such as World Wide Consortium (W3C) and OASIS are in the process of devising such standards.

Many proprietary SOA tools and frameworks have also been produced for the development of WSs 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 [18].

IV. SOA Promise and Benefits

SOA offers better opportunities for enterprise application integration with the additional advantages of reduction in costs, ease of maintenance and improvement in flexibility and scalability. SOA allows enterprises and their IT systems to be more agile to the changes in the business and the environment.

It provides an opportunity to achieve broad-scale interoperability while offering flexibility to adapt to changing technologies. If implemented correctly, its technical characteristics translate directly into real bottom-line benefits. The main characteristics include:

- Loosely coupled architecture
- Modular approach
- Non-intrusive nature
- Universality of standards.

Various benefits derived from the above can be summarized as follows [9], [10]:

- Loosely coupled applications and location transparency – this allows enterprises to plug in new services or upgrade existing services with relative ease [4].
- Seamless connectivity of applications and interoperability – this provides opportunities to increase business agility and the ability to respond *on demand.*
- Alignment of IT around the needs of the business – this results in IT as the enabling technology to provide added value to business operations.
- Increased business agility, capturing new channels of business – this provides flexibility, ease of access and increased customer satisfaction.
- Enhanced reuse of existing assets and applications – this helps to reduced costs, reduced development time and a reduction of time to market
- Relatively easy integration of legacy systems – this promotes interoperability and efficient use of resources and existing facilities.
- Process-centric architecture and flexibility of approach – this allows a process driven approach, which is a worthwhile aim to achieve.
- Parallel and independent applications development – this is due to the reuse of services and ability to withdraw or plug in new developments with considerable ease.
- Better scalability and graceful evolutionary changes - this is due to the reuse of services and ability to develop applications independently and in parallel.

Other associated benefits include:

- Reduced costs of integration and change management.
- Automatic derivation of process metrics and hence reduction of process metrics costs.
- Easier and more effective systems maintenance.
- Reduced business risk and exposure in spite of increased business visibility.
- Reduced vendor lock-in.

The SOA represents a new way of developing systems - it promotes a shift from writing software to assembling and integrating services. A natural consequence is that systems are no longer implemented as single monolithic structures but as a series of component parts that work together to deliver whatever functionality is required. It is an effective strategy for integrating enterprise–wide applications and legacy systems. Underlying platform implementation becomes irrelevant as standard interfaces and message exchange patterns provide integration, both within and across enterprises. The goal is to build a flexible infrastructure to abridge applications more easily and to solve business challenges quickly. However, to support the goal of SOA, the infrastructure must support flexibility, heterogeneity, distributed development and management [9].

V. SOA Limitations and Inherent 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.

Obviously, it is desirable 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. Hohpe [5] mentions the following problems with respect to the existing distributed architectures:

- Vendor lock-ins: as many architectures are based on proprietary protocols and implementations.
- Tight coupling: as distributed architectures typically link components directly to one another.
- Complexity: as the interactions between objects are often rich and complex.
- Connectivity: as majority of distributed architectures do not work over wide-area, intermittent networks.

SOA also requires a large upfront investment by way of technology, development and staff deployment. It may cost a great deal and the Return on Investment (ROI) could take a long time to materialize. 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 negative implications by way of response time and overall machine load
- It is entirely possible, at times, 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].

Technology risk of SOA is particularly challenging due to the following factors [19]:

- Early adoption – especially when the technology is new and evolving,
- Organizational changes within the enterprise – to accommodate a new way of thinking,
- Architecture – this is enterprise wide and thus encompasses heterogeneous systems,
- Quality assurance – this is particularly difficult as services are distributed and ownership is often unclear.

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.

Issues, inherent due to the very nature of service-orientation, can be summarized 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 optimized 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 summarized 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 a 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/received in a timely manner [17].
- 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 organizations 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 organizations. 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 and thus security becomes an issue [13]. 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. Thus, service management becomes a real issue.
- Choosing a vendor: Knorr and Rist [2] recommend a multi-vendor solution to fully benefit from the flexible nature of SOA.
- Training: It takes time to learn and use new technologies. There are too many technologies and not enough expertise available in the market. For an organization adopting this architecture, thorough understanding of the underlying technologies remains critical [8].
- Adoption: staff's reluctance to embrace a new technology is always an issue that management needs to consider carefully.
- 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.

Given the challenges and issues, as mentioned above, Roch [19] recommends the following steps to mitigate, at least, some of these risks:

- Develop a SOA Programme Charter based on long term business objectives.

- Determine a SOA strategy and roadmap based on business values and risks.
- Examine the architectures and methodology in use and adjust for SOA.
- Establish a training programme to acquire the necessarily required skills.
- Develop SOA Quality Assurance policies and procedures.
- Involve operations support early and deploy monitoring and management tools.
- Establish a clear governance policy, especially for the management of services.
- Adopt evolutionary approach.

Moving to SOA is not an easy transition. It requires a shift in how we compose service-based applications while maximizing 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.

VI. Conclusion

SOA provides a new way of developing and integrating enterprise applications, however, 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, training in new technology, huge initial investment, unreliability of Internet protocols, evolving standards and the newness of the approach.

The bandwagon for SOA is rolling and many companies are already jumping on it. Enterprises, thinking of moving in this direction, 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. They need to be aware of the difficulties and the inherent issues, which are many.

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. There is no doubt that enterprises will need to embrace this new paradigm: if not now, then in very near future.

References

[1] I. Cartright and E Doemenburg, "Time to jump on the bandwagon" in IT Now, British Computer Society, UK, May 2006.

[2] E. Knorr and O. Rist, "10 steps to SOA" in Info World – San Mateo, vol. 27, issue 45, Nov 2005.

[3] D. Barnes, "The service oriented architecture: more than just another TLA?". British Computer Society, UK. Available: www.bcs.org/server.php?show= ConWebDoc.3041.

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

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

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

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

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

[9] Sonic Software Solutions, Service oriented architecture. Available: www.sonicsoftware.com/solutions/service_oriented_architecture/index.ssp

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

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

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

[13] Wikipedia, "Service-oriented architecture", Available: 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] C. Saran, "SOA will fail without governance: warns Gartner", *Computer Weekly, UK*, 12 Sept 2006.

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

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

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

[18] Z. Mahmood, "Service oriented architecture: tools and technologies", *Proc 11th WSEAS Int. Conference,* Crete, Greece, July 2007.

[19] E. Roch, "Service oriented architecture and technology". Available: http://blogs.ittoolbox.com/eai/business/archives/soa-benefits-challenges-and-risk-mitigation-8075.

Zaigham Mahmood is a Senior Lecturer, and Scheme Leader for undergraduate programmes, in the School of Computing, University of Derby, UK. He has an MSc in Mathematics, a specialization in Computer Science and a PhD in Modeling of Phase Equilibria. He is also a Chartered Engineer (Engineering Council, UK) and a Chartered Information Technology Professional (British Computer Society, UK).

Dr Mahmood has more than 35 papers published in proceedings of international conferences or journals. His research interests are in the areas of software engineering, project management, software metrics and process improvement.