INTERNATIONAL JOURNAL OF COMPUTERS
Issue 3, Volume 1, 2007

Enterprise Application Integration based on Service Oriented Architecture

Zaigham Mahmood

***Abstract*—**Enterprises have invested heavily in large-scale applications software to run their services and business functions. The infrastructure used is often heterogeneous across a number of platforms, operating systems and languages and, thus, there is 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 while leveraging existing IT investment as well as provide flexibility and on-demand services. In this context, Service Oriented Architecture (SOA) is emerging as an attractive architectural style for developing and integrating enterprise applications. SOA promises a better alignment of IT with business, seamless integration of business applications and reduced costs of development and maintenance. Evidence suggests that large enterprises are moving towards this new paradigm. In this paper, we introduce the SOA approach, present the benefits and challenges it offers and provide guidance with respect to enterprise application integration and implementation. The objective is to provide some useful background information for enterprises that wish to embark on the road to application integration via the SOA.

***Keywords*—**Service oriented architecture, SOA, enterprise applications integration, EAI, XML.

I. Introduction

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 used is often heterogeneous across a number of platforms, operating systems and languages. There is, thus, a huge duplication of functionality and services resulting in a waste of valuable resources and poor response times. Increasingly, the businesses, especially those in the global markets, are being asked to deliver improved functionality of services, while leveraging existing IT investment, as well as provide the following:

- Business agility
- Meeting customer demands
- Reduced time to market
- Continuous process improvement
- New channels of business
- Business architecture that is *organic* in nature [1].

Manuscript received January 31, 2007; Revised May 19, 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).

One solution is to develop architectures that allow easy integration of the existing and new enterprise applications. However, the integration solutions are often proprietary which present issues of inoperability because of vendor lock-ins, tight coupling, complexity of services and issues of connectivity [2].

The Web Services (WS) technology and Service Oriented Architecture (SOA) offer better opportunities for enterprise application integration (EAI) 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 [1] and provide opportunities to 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.

In the rest of this paper, we first establish the need for SOA and mention the potential benefits that SOA aims to achieve. In sections III and IV, we outline the SOA framework and technologies and discuss the limitations and inherent issues. Section V presents suggestions for integration of enterprise applications and implementation of the SOA paradigm. In the last section, we present summary and conclusions.

II. SOA And Its Promise

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 [3] providing a way of sharing business functions in a widespread and flexible manner. Knorr and Rist [4] 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* [5]. This architecture aims to provide enterprise 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 EAI to provide a closer alignment between a business and its IT systems.

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 development and integration.
- Reduced vendor lock-ins.

III. SOA Framework 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 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 and a service is an exposed piece of functionality.

The World Wide Web Consortium (W3C) Web Services Architecture Working Group defines SOA in terms of services, characterized by the following properties [9]:

- Logical view: A service is an abstract, logical view of a program, database or a business process defined in terms of what it does, typically carrying out a business level operation.
- Message orientation: A service is formally defined in terms of the messages exchanged between service providers and service *clients* (refer to Fig. 1).
- Description oriented: A service is described by a machine process-able meta data.
- Granularity: A service having a small number of operations with relatively large and complex messages.
- Network orientation: Each service primarily designed to be used over a network.
- Platform neutral: Messages are sent in a platform neutral, standardized format, over the network.

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 are dynamically located
- 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, administrative functions, data logging, exception handling, as well as registration and discovery,
- 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 relation to an enterprise information architectural framework, considering a business from the viewpoints of enterprise and technology, SOA can be viewed in terms of 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 realization of services in the business architectural model as a combination of smaller much finer gained services.
- Implementation architectural model: this refers to the system realized in a certain manner using certain software and hardware systems.

At the highest level of abstraction, SOA uses a *publish-find-bind-execute* paradigm as shown in Fig. 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.

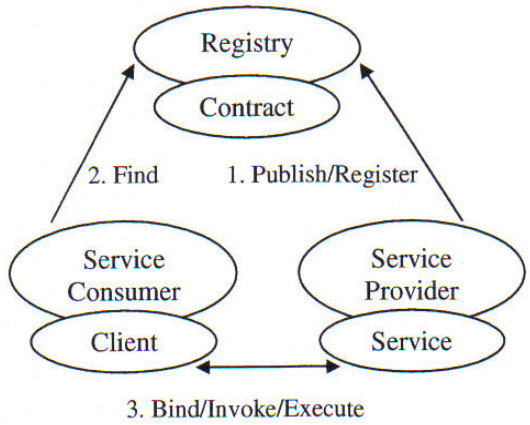

Fig. 1 Publish-Find-Bind-Execute paradigm

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, 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 the services. However, for the SOA to become a mainstream IT practice, other standards such as those to do with security and services management need to be added. Such standards, referred to as WS-*, are already emerging and organisations such as 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 [2], [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 products from vendors such as BEA, Eclipse, IBM and CapeClear, refer to Mahmood [20].

IV. SOA Limitations And Inherent 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 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 summarized 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 optimized 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 a rework of existing code to conform to new 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 messages are delivered and 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 (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.

V. SOA Implementation For EAI

Developing services as software components is not a new idea and, therefore, not too complex, however, implementing SOA is not an easy task. The entire process requires a shift in how we compose, sequence and integrate service-based applications while maximizing 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].

In its simplest form, SOA involves exposing reusable software modules as Web Services, which are combined in a loosely coupled fashion to create composite business applications. However, SOA requires building systems at a business level, not just at the IT level – as IT becomes an enabling technology to further the cause of the enterprise. 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.

There are many aspects to SOA that must be addressed in order to succeed. Perhaps the most important is a clear understanding that SOA is not just about technology - it is also an IT strategy to support business transformation and, as such, SOA has extensive organizational, procedural, and process implications. It should also be noted that SOA is less concerned with integration issues and more with management and governance issues. Whereas, 30% of SOA is about development, the rest is about governance and managing infrastructure [24].

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 organizing a service directory, to act as 'yellow pages' based on open standards (e.g. UDDI) where each service has a clear *contract*.
- Integration: of services with applications and other services, including data transmission services, reliable message delivery mechanism etc.
- Orchestration: to 'sequence' the required services to fulfill 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 on-demand availability – a requirement of an agile system.
- 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 process of development and execution and analysis with respect to process improvement.

As for the deployment of services and to realize the benefits that SOA offers, enterprises need a flexible infrastructure capable of supporting dynamic operations with flexibility, inherent throughout. In this respect, provision of a distributed hardware systems in the form of Grid Computing provides a sensible approach. Grid computing allows the sharing of resources of disparate computer systems (including storage) with the following benefits [21]:

- Provides a framework for exploiting unutilized resources,
- Allows resources to be geographically widely dispersed,
- Provides a balanced resource utilization,
- Allows better viewing of usage patterns for better planning with respect to further acquisition or upgrading of systems,
- Allows dynamic allocation of resources,
- Provides a more uniform interoperability among heterogeneous grid participants.

With a Grid computing in place, it can be usefully employed for SOA initiatives, as it will resolve many of the issues of security, deployment and management of services as well as overall central administration [22]. Just as an SOA allows customers to separate applications from services, grids allow customers to separate both applications and services from the infrastructure and systems resources [23].

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 organizations can use the following strategy [2]:

- Start with the main business processes that span multiple business units.
- Identify services to support business processes.
- Identify operations that can be exposed as services to support 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 costs of development.

It is also important that companies begin their efforts with small pilot projects in order to better understand the potential of the technology and the processes.

VI. Conclusion

EAI, using SOA, is an approach that promises numerous benefits, however, SOA requires enterprises to determine the correct 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 and 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 it. 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, characteristics of SOA, the potential benefits that SOA promises as well as the inherent issues and limitations are discussed. Relevant framework, associated technologies and guidance on building and implementing a SOA have also been discussed. The objective is to provide some useful information for enterprises wishing to embark on the road to business application integration using SOA.

References

[1] 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
[2] G. Hohpe, "Developing Software in a service-oriented world", *Whitepaper, ThoughtWorks Inc.*, Jan 2005.
[3] I. Cartright and E. Doemenburg, "Time to jump on the bandwagon" in IT Now, British Computer Society, UK, May 2006.
[4] E. Knorr and O. Rist, "10 steps to SOA" in Info World – San Mateo, vol. 27, issue 45, Nov 2005.
[5] D. Barnes, "The service oriented architecture: more than just another TLA?", British Computer Society, Available: www.bcs.org/ server.php?show= ConWebDoc.3041.
[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: sonicsoftware.com/solutions/service_oriented_architecture/index.ssp
[10] B. Johnson, "The benefits of service oriented architecture", *Objectsharp Consulting*, Available: 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/~olga ce/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. Stojanovic, "A Dahanayake and H Sol, Agile Modelling and Design of SOA". Available: www.cs.ucl.ac.uk/staff/ g.piccinelli/eooows/ documents/paper-stojanovic.pdf
[19] T. Erl, "Core principles for service-oriented architectures", Aug. 2005. Available: Looselycoupled.com/opinion/2005/erl-core-infr0815. html
[20] Z. Mahmood, "Service oriented architecture: tools and technologies", *Proc 11th WSEAS Int. Conference*, Crete, Greece, July 2007.
[21] V Berstis, "Fundamentals of Grid computing", *Redbooks Paper, IBM*, 2002. Available: www.ibm.com/redbooks
[22] A. Sedighi, "Enterprise SOA meets Grid". Available: www.gridtoday.com/grid/ 695457.html
[23] E. Kaurpas, "Grid Computing: past, present and future: an innovative perspective", *IBM*, June 2006.
[24] H. Beckett, "Putting the SOA pieces together", Computer Weekly, UK, 17 April 2007.

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 35 papers published in proceedings of international conferences and journals. He is also a member of advisory and editorial boards of several journals and international conferences. His research interests are in the areas of software engineering, project management, software metrics and process improvement.