Prerequisites for Successful SOA Adoption

George Feuerlicht
University of Technology, Sydney
jiri@it.uts.edu.au

1. INTRODUCTION

The adoption of SOA (Service Oriented Architecture) has gained momentum in the past two years, and the predictions are for further rapid uptake of SOA and Web Services. In a recent report, Gartner estimates that SOA will be used in more than 50% of new, mission-critical applications designed in 2007, and in more than 80% by 2010. At the same time, there are indications that despite the overwhelming industry agreement on adoption of SOA, a majority of IT leaders remain uncertain about the costs, and the technical and the organizational prerequisites for SOA. There are also signs that not all SOA projects are achieving the stated objectives of improved reuse and agility leading to improvements of ROI (Return on Investments), and some projects fail altogether. According to Gartner, SOA is approaching the "Trough of Disillusionment" on the Gartner Hype Cycle. Some of the current disillusionment with SOA relates to unrealistic expectations caused by confusion about the scope of the problems that SOA is capable of addressing. The concept of SOA has become *overloaded*, with some vendors and analysts promoting SOA as a "silver bullet" solution to most IT problems, claiming that SOA is capable of delivering improved agility, faster time to market, easy integration, improved reuse, better scalability, while at the same time reducing implementation risks and costs. Similar claims were made a decade ago for component-based architectures, and earlier for client/server causing wide-spread skepticism about such claims. It is becoming clear that SOA benefits cannot be achieved overnight simply by switching over to a new technology platform, and that successful SOA adoption requires major change in the implementation and management of business processes across the organization and between business partners and careful management of the transition.

1.1 What is SOA?

Lack of a clear SOA definition is a part of the problem. SOA is a set of evolving architectural concepts that together form a blueprint for a new enterprise computing architecture. SOA involves the concept of a service as a basic building block of distributed applications and an Enterprise Service Bus (ESB) - a software infrastructure that enables SOA by acting as a middleware layer that facilitates the interactions between services, and provides a run-time environment for service-oriented applications. ESB (or some other type of SOA middleware) manages service deployment and execution, and controls access to services based on security and QoS (Quality of Service) attributes. SOA promotes loose coupling and document-based communication using coarse-grained, asynchronous messaging. Many SOA concepts have been adapted from earlier generations of distributed system technologies. SOA has benefited from over twenty years of distributed systems research and development that includes J2EE and .Net component architectures. SOA represents the next stage in the evolution of distributed computing, and the benefits of implementing SOA in organizations are likely to be

delivered incrementally. Gradual adoption of SOA and its coexistence component-based architectures and legacy applications is a good strategy minimizes the risks associated with SOA projects.

The emergence of SOA is to a large extent the result of an unprecedented lev standardization based around Web Services. The SOA standardization s extends from low level protocols such a SOAP up to the level of business pro execution language BPEL. However, agreement on technical standards alone c not ensure that the business benefits of SOA will be fully realized. As with any c technology, Web Services can be designed and deployed in many different w and with variable results. SOA best practices are still being formulated evaluated in large-scale implementation projects, and many SOA features are the subject of ongoing research.

1.2 What determines the success of SOA projects?

Successful SOA adoption requires that the organization reaches a level of mat in a number of important respects that include technical readiness, alignmer organizational business processes and services, and the definition of the governance model. Organizations need to develop a roadmap for transition to that addresses the "big picture" issues of SOA adoption including SOA governa implementation of enterprise SOA infrastructure based on accepted standards, the transformation of legacy applications. At the same time, organizations nee acquire detailed technical knowledge and skills, and introduce compreher methodology to support the entire SDLC (Systems Development Life-Cycle service-oriented applications. It is becoming clear that SOA adoption needs t conducted in several stages to avoid the risks associated with the "big b approach, initially identifying projects that are likely to deliver good retur investment within a relatively short period of time (e.g. 3 months). However, it be dangerous to assume that successful small-scale pilot projects conducted u controlled conditions by a team of highly skilled external consultants can guara a successful SOA roll-out across the entire organizations. For example, se reuse cannot be satisfactorily evaluated in the context of a small-scale pilot pr and needs to be addressed by fully analyzing requirements across organization.

2. SOA SYSTEMS DEVELOPMENT LIFE-CYCLE SUPPORT

As noted above, a key success factor for SOA adoption is the introduction comprehensive methodology that supports the development of new applicatio well as the integration of existing, legacy systems. Implementation of applications requires reliable methodologies and tools, covering requiren analysis and modeling, service design, and implementation phases of the SDL service-oriented applications. For each SDLC phase, project roles corresponding skills need to be specified.

2.1 Service analysis and modeling

Service analysis and modeling has to support business-level conceptual modeling of services and their relationship to business processes, capturing both functional and non-functional requirements. Business Process Modeling (BPM) and Object-Oriented Analysis and Design (OOAD) are well established techniques that have been applied to SOA, but experience gained from SOA implementation projects suggests that such methods only partially support SOA architectural patterns. One key difference concerns the service contract definition. Component-based approaches define the contract purely from functional point of view as the signatures of interface operations and the corresponding pre- and post-conditions. SOA introduces non-functional requirements that include QoS (Quality of Service) and related considerations. Another key difference is that the level of abstraction used for modeling services is higher than normally adopted for component-based systems, and often involves services that represent complex business processes. Identifying the most appropriate level of service granularity during analysis and design phases represents a major challenge to developers of service-oriented applications, and the topic of optimizing service granularity is an active research area at present. Service modeling methodologies are still evolving and opinions differ about the precise nature of the relationship between services and business processes on one hand, and services and components on the other. While most experts recognize the importance of BPM for SOA, the precise nature of the relationship between business processes and services, and the mapping of business processes captured using notation such as BPMN (Business Process Modeling Notation) or UBL (Universal Business Language) into executable languages such as BPEL is still under investigation.

2.2 Service Design

Service design is concerned with the transformation of service models produced during analysis into a set of design specifications and artifacts such as service interface specifications and composition/choreography workflows that satisfy specific application requirements. The transition towards service-oriented computing necessitates re-evaluation of design methodologies that are currently used in the construction of enterprise applications. Service design needs to determine what constitutes a service and its operations, and make decisions about the granularity of services based on previously developed service models. In order to achieve the benefits of reuse, extendibility and responsiveness to new business requirements, services must be specified at the correct level of abstraction and granularity. Correct design is crucial as using coarse-granularity services inhibits reuse, and fine-grained services cause higher network overheads and more complex interactions dialogues. Furthermore, coarse-grained service operations are less flexible and more difficult to evolve as a number of business functions are typically combined and implemented as a single operation. Frequently, such services exhibit overlapping functionality and necessitate extensive modifications when the requirements change. Opinions differ on how an optimal level of service granularity should be determined.

2.3 SOA Development Tools and Environments

In addition to comprehensive modeling and design methodologies, developers c service oriented applications need mature and stable development and deploymer platforms. Transition to SOA cannot take place in the absence of applicatio development environments that provide comprehensive support for the variou SOA life-cycle stages. Such IDEs (Integrated Development Environments) mu: support industry standard languages and interfaces (i.e. WSDL, BPEL, Java, etc and enhance developer productivity by minimizing low-level coding and usin declarative methods whenever possible. The Service Component Architectur (SCA) is an industry effort by BEA, IBM and Oracle aiming to provide a model f the assembly of business solutions from a collection of individual services. Th emphasis is on increasing the level of abstraction and focusing on the busines problem, minimizing the direct use of low-level APIs, and access methods. Give the industry momentum behind SCA it is likely that many vendors will incorpora support for SCA based development into their products.

3. CONCLUSIONS

There is now little doubt that service-oriented computing will play a major role enterprise computing in the future. Equally, it is evident that SOA cannot instant solve all enterprise computing issues. A key benefit of SOA is that it enables close alignment of IT architecture with the business requirements and at the sam time facilitates high-levels of business process automation. While SC implementations can deliver short-term, incremental benefits in integration projec by wrapping existing application components as services, the main, long-ter benefit of SOA is that it enables organizations to participate in the emerging wor of service-oriented computing that involves sourcing application services externall and participating in domain-wide service-oriented applications.

The benefits of SOA can be only fully realized if reliable technological solutio supported by comprehensive SDLC methodologies are used to develop and depl service-oriented applications. Another key determinant for the successful adoptic of this new approach is the level of the skills and knowledge of IT professiona involved with SOA projects. People involved with introducing SOA in organizations need to be able to make informed decisions about technolo selection, the timing of adoption of various Web Services standards and solution and successfully manage large-scale SOA projects.

The 15th International Conference Systems Integration 2007 includes a SC Workshop, presented on Sunday, 10 June, 2006 with a focus on the late developments in this area. Presentations by international experts inclu papers on SOA directions, Web Services standardization, and SC implementation case studies. The workshop will provide an opportunity discuss SOA implementation issues and share experiences with SOA a Web Services development.