The Business Value Propositions of Service-Oriented Architectures

ZELJKO PANIAN
The Graduate School for Economics and Business
University of Zagreb
J.F. Kennedy Sq. 6
CROATIA
http://www.informatika.efzg.hr

Abstract: - Service-Oriented Architecture (SOA) concept provides the software industry with a powerful business value extension not available with hardware solutions – a model that supports flexible and dynamic assembly of business solutions at runtime from a variety of locations. In addition to addressing technical complexities, business drivers for an SOA framework model include the capability to significantly improve the time-to-market delivery of the business solution and require less technical resources to deliver it. A primary goal of implementing Service-Oriented Architecture is to simplify development and implementation of new applications and capabilities through aggregating many low-level tasks into higher-level services. A secondary goal of SOA is to enable any application or process to utilize the services of any other application or process, over any means of interconnection. The universality of SOA can help enterprises use their existing systems in new ways, for new purposes, for greater competitive advantage, efficiency, and return on investment.

Key-Words: - Service, Enterprise Service Bus, Web Services, Service-Oriented Architecture, business process, business value proposition**,** implementation methodology, Business Process Execution Language (BPEL)

1 Introduction

A key business goal is to grow revenues while concurrently lowering operating costs. Since the early 1980s, the personal computer industry has fueled significant productivity growth and reduced operating costs for businesses mainly due to their engineering model of assembling a wide variety of systems from commercial off-the-shelf (COTS) pre-built components.

In this same time period, the software industry continued to evolve engineering models in an attempt to mimic the successful hardware industry approach. Service-Oriented Architecture (SOA) represents the latest and closest step in this evolutionary process.

With self-contained, modular, loosely coupled, location, platform and protocol independent attributes of a service-oriented approach, software solutions can obtain the key capability of the successful hardware engineering model – commercial off-the-shelf solution assembly.

In addition to realizing the COTS vision, Service-Oriented Architecture concept provides the software industry with a powerful business value extension not available with hardware solutions – a model that supports flexible and dynamic assembly of business solutions at runtime from a variety of locations.

To realize the vision of dynamic and flexible assembly of commercial off-the-shelf business solutions, the technical complexities of constructing, assembling, and controlling solutions have to be addressed. As will be described later in this paper, these complexities are key technical drivers for an SOA framework model.

In addition to addressing these technical complexities, business drivers for an SOA framework model include the capability to significantly improve the time-to-market delivery of the business solution and require less technical resources to deliver it.

2 Service-Oriented Architecture Basics

OASIS, a non-profit global consortium defines SOA as “an architectural style whose goal is to achieve loose coupling among interacting software agents or services” [1].

According to the same source, service is “a unit of work performed by a service provider to achieve desired end results for a service consumer”. Both

provider and consumer are roles layered by software agents on behalf of their owners.

Unfortunately, these definitions are often diluted due to the marketing efforts of many software companies who attempt to define SOA through any one number of technologies, such as Enterprise Service Bus (ESB) [2, 3], Business Process Execution Language (BPEL) [4], or Web Services [5, 6].

The fact is that most of these technologies indeed contribute to an SOA, but an SOA is far more extensive and general than any one of these standards.

To further break down the definition, it is important to understand that services come in different types; coarse-grained and fine-grained. The granularity is based on how much of a service is exposed.

Coarse-grained services are constructed from lower-level services, components, and objects that are intelligently structured to meet specific business needs. Fine-grained services provide a small amount of business process usefulness, such as basic data access or simple data conversion.

SOA is a model based upon the loose coupling of services of optimal granularity to support the desired business processes [7]. In fact, taking a look back at the evolution of software development up to SOA, it is easy to see the challenges developers have faced over the past several years and to observe the solutions that have been proposed to solve the problem of distributing and sharing the logic leading up to SOA.

The principles of structured and modular design, followed by object-oriented programming, have addressed the challenge of effectively distributing and re-using code to better facilitate the development of custom business applications. SOA extends these concepts out to the service level to make such services available to those outside of IT.

3 Goals of Deploying Service-Oriented Architectures

A primary goal of implementing Service-Oriented Architecture is to simplify development and implementation of new applications and capabilities through aggregating many low-level tasks into higher-level services. To use human analogy, the heart is a discrete system comprised of smaller, specialized systems, such as valves, arteries, and veins. Lungs, the circulatory system, and brain are other vital human macrosystems. Ultimately, the body's major macrosystems – respiratory system, circulatory system, nervous system – collaborate to support global processes such as respiration, speech, and movement.

Within each level of aggregation, the microsystems function independently; however, each successively higher level sees the lower-level functions as ubiquitous and available to it through a common interface. The heart does not worry about what the lungs do or how they do it; when the heart needs oxygen, it simply extracts it from an artery without knowing how it is supplied.

An SOA is similar. All electronic business processes are implemented at the lowest levels as specialized functions designed to perform specific individual tasks. Low-level services are designed like text messaging – as short, to-the-point interactions. These low level services created and maintained by people who have expertise required to weave them together to produce desired business effect.

A service does not necessarily know about other services or how they perform. For example, when a financial reporting application needs data, it simply asks for it from a software service without knowing or caring from where or how the data is supplied. And at higher level of aggregation, neither does a Web application, portal, or application system worry about what other applications do.

A secondary goal of SOAs is to enable any application or process to utilize the services of any other application or process, over any means of interconnection. Enterprise business processes rely on wide variety of message formats and transports to communicate within and between systems.

A unique message format, combined with the communication transport over which it travels, is called a service channel. Service channels define unique language and communication characteristics needed to enable disparate systems to collaborate. In SOA, any service should be usable over any type of service channel, rather then being confined by unique or proprietary combinations of message format and channel. SOA can aggregate a business enterprise's service-channel architecture into a single, extensible service-channel implementation, enabling any service to operate across any channel without having to implement multiple gateways or trading partner agreement managers.

The universality of SOA can help enterprises use their existing systems in new ways, for new purposes, for greater competitive advantage, efficiency, and return on investment.

The SOA system is schematically presented on Fig. 1.

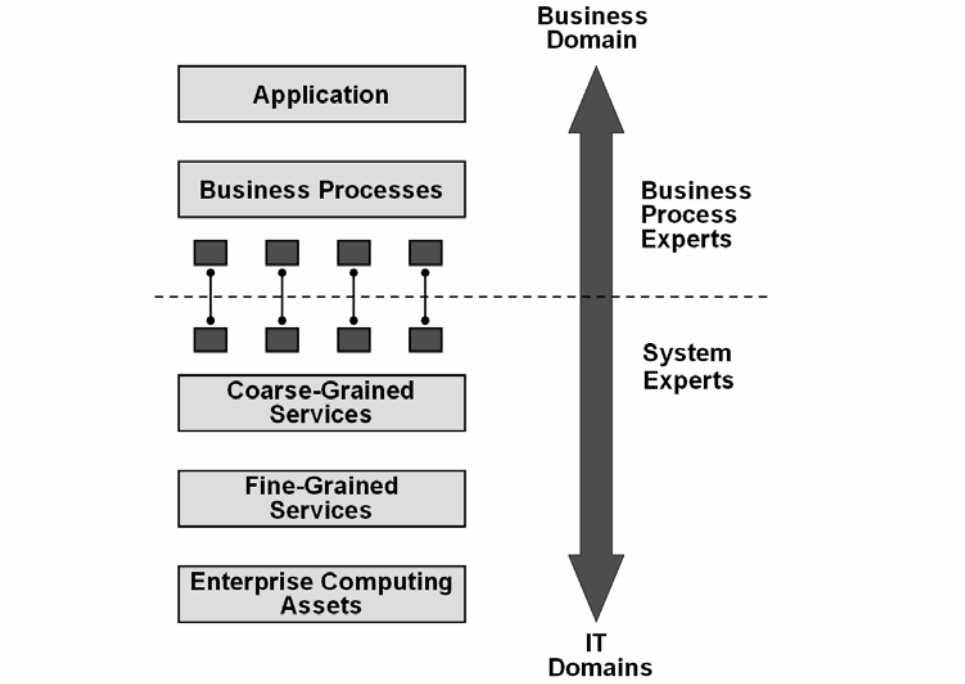

Fig. 1 – The SOA System

4 The Importance of Service-Oriented Architecture

With a focus on revenue growth combined with lower operating costs, enterprises are continually challenged by the need to upgrade and create custom applications across the enterprise to accommodate even new and more demanding business requirements.

The urgency to upgrade applications to reusable services, both internally and externally, is fueled by the demand to provide integrated access to information for anyone, from anywhere and anytime.

For example, consider the Company X that has a new set of upgrade requests from the business for their customer support applications. These new requirements mandate that outside sales, the call center and online customers gain access to the same set of corporate information. Outside sales will require remote access to purchase history, support, inventory, and payment details. Customer service requires access to the same data for taking customer inquires, but requires a "rich client" interface. The new requirements also include a Web-based self-service application where online customers can check order status, track shipments, and view payment verification.

This example depicts a typical enterprise in need for Service-Oriented Architecture. Enterprises are quickly adopting strategies to leverage real-time services both internally and externally in an effort to increase their bottom line. A SOA platform enables IT to respond promptly, meet the changing needs of the business, and extend applications by adjusting the model without re-writing the existing code.

We can extend the example above with Company X to include a requirement for additional access to extended business data at Company Y. Company Y is a multi-channel supplier to Company X. The new requirements mandate support for Company X's customers and employees to access order and shipping details purchased online from Company X's Web site, but supplied by Company Y.

Requirements for extending custom applications beyond the enterprise are more common in today's business enterprise. An SOA platform can satisfy this requirement without use of any additional technology. Business users connect to the SOA platform to select which partners they will interact with, and how they will interact. The platform automates inter-business communication to the full extent by placing it in hands of the business users.

5 Service-Oriented Architecture Implementation Methodology

Moving to Service-Oriented Architecture requires a strategic commitment to create a more flexible IT system that maps closely with business processes, instead of trying to retrofit business requirements into technology decisions. Once the commitment is made, the path to SOA can either be very difficult or extremely straightforward based on technologies and practices.

The first version of any new implementation is always the most critical, difficult and expensive step until optimal operation is achieved. The challenges of training, evolving best practices, collaboration, and performance and tuning are expected. The right methodology applied can ensure that even though these challenges exist when moving to SOA, significant complexity and expense are taken out of the process through productivity, standards-based technology and the industry's lowest Total Cost of Ownership.

The implementation should focus on SOA from the ground up to facilitate not only the design and build of an SOA, but also the control or governance, deployment and management of that SOA within the enterprise and beyond. By enabling the full lifecycle of services, such a methodology can reduce the cost and decrease the time to value for an SOA.

5.1 Constructing and Modeling the custom SOA Services

The need to retrain or upgrade existing skill sets will depend on the platform selected for the SOA. Many SOA platforms require extensive engineering and knowledge of specific programming languages in order to extend custom services and evolve SOA throughout the enterprise. Most of today's development platforms, which are re-branding themselves as Service-Oriented Architecture, require serious and detailed knowledge of .NET, J2EE, or other programming languages.

The ability for these platforms to enable business service development, deployment and management is, however, very limited. Services developed using these frameworks, otherwise known as Integrated Development Environments (IDEs), require extensive engineering expertise, not only for development but also for distribution. This often results in the retraining or rehiring of IT staff. But, the platform is needed that could be built for SOA from the ground up, which is a key differentiator when evaluating an SOA platform.

Early best practices for SOA strongly support a model-driven approach to composite application development, modularity, flexibility, and extensibility. Application modeling is especially useful to SOA, as it allows for architects to evolve their solutions in flexible ways and enable companies to explain, define, manipulate and store processes, as they exist – apart from the specific technology needed to support them.

Many codeless tools and claims have surfaced over the years and many of them have failed when it came to proving their promises. The evolution of such claims has gotten closer to the mark over time, and current technology and design metaphors have made codeless development a reality.

Model-driven approach to Service-Oriented Architecture enables technologists to focus on building business services. By eliminating the time-consuming process of manually writing code, companies can drastically accelerate their solution development.

5.2 Assembling SOA Components

Some of the confusion in the acceptance and adoption of SOA has been the misconception that Web Services and Business Process Execution Language (BPEL) are the key components of building out an SOA. Both have played a large part in the evolution of business and application development and certainly have their large applicability. However, they are merely components in successful SOA.

Services within the SOA can be created in many different fashions – some in Java, some in C#, and some in C or C++. Deploying these services as Web Services would be indeed well suited for interoperability, as they are by nature heterogeneous, interoperable, and loosely coupled. However, when building end-user applications or high volume processes, Web Services may not be the best choice.

Furthermore, Web Services are expensive in both processing and bandwidth required. On average, a Web Service invocation is nearly 10 times larger than the binary form of such an object interaction [8]. Using eXtensible Markup Language (XML) and Simple Object Access Protocol (SOAP) requires huge chunks of bandwidth when dealing with large numbers of transactions, as both the SOAP transport and all data are passed as XML, which is expressed as raw text.

BPEL, in theory, is closer to SOA today and has been a good way to link and assemble some business processes although it is not an enterprise platform for deploying high volume custom

business applications. BPEL only enables the calling of processes, but does not allow for new processes to be developed.

Business Process Execution Language can be an optional layer of SOA, but BPEL alone will not deliver a Service-Oriented Architecture, as it relies entirely on Web Services and XML. Some of the early problems with BPEL have been difficulties in adhering to appropriate standards [9].

Considerable customization is required to implement BPEL. Standards are often lost in the process, and many of the services end up becoming proprietary solutions, thereby losing the ability to be quickly extensible as the business requirements change.

These are challenges that can be solved by implementing a distributed infrastructure comprised of Web Services, Java, and other native services. Such architecture generates the ability to orchestrate and assemble standards-based custom business services that can be deployed and reused across the enterprise, enabling the sustainable alignment of IT and business requirements.

One of the most important practices for building SOA is planning for the future as business needs change. Services needed to be leveraged and extended to immediately respond, without the need to reinvent and reconfigure each process.

5.3 SOA Performance Control and Monitoring

Some services built today will become a part of larger service implementations in the future as Enterprise SOA [10]. As the SOA grows into several thousand services, manageability must be a key consideration early on.

The main value propositions behind SOA are:

- reuse,
- efficient development,
- simplified maintenance,
- enabling the sustainable alignment of IT, and
- portability.

These will provide a flexible evolution enabling more advanced business processes.

However, managing those processes is a significant challenge. The problem domain for SOA control, i.e. managing service development and deployment, includes delivery levels and solution lifecycles.

In terms of delivery, control is required at the application, systems and infrastructure levels. As SOA implementations expand, different levels of service management will be required in order to deploy, distribute, extend, and monitor the SOA.

The SOA platform market contains many technologies that address management and infrastructure monitoring, however these tools do not address the provisioning of service development, integration, and distribution.

As services move through development, such as quality assurance and delivery, maintaining consistency and quality along the way is a key component. The release cycle is often forgotten element in many projects, too.

Once a project is released, a different set of challenges arises for updating released versions of the software due to the many dependencies that exist. With SOA, the challenge is even greater as the goal of SOA is to deploy services only once and access them from anywhere.

Suppose a few thousands services being deployed to many different servers throughout the enterprise. The URL and location of those services will differ from that of existing production services, multiplied by the number of services being released. The easiest solution for most would be to redeploy the service many locations. This defeats the value of SOA and creates more complexities around service versioning and maintenance.

The problem of effective release management and deployment of services can, however, be solved. The platform used should naturally maintain location independence and allow services to progress through each development phase without the need to duplicate to the same service in multiple locations.

The platform should also provide service management at the application delivery level, enabling discrete control over business services as they move across different business domains (e.g., Human Resources, Finance, Marketing, Customer Service, etc.).

6 The Future Requirements

In tomorrow's SOA model, many of the services that will become a part of the solution are already available and deployed to production systems. In order to leverage these services without having an affect on production environment, it will be necessary to browse the environment, version service integration and deployment.

In addition, service discovery and introspection properties such as location, department, author, last access and type of service are needed, which will enable and simplify reuse from the start. This

functionality would allow administrators to visually model and configure the best possible scenario for application deployment.

In the future, a Web-based console should be also developed to display more granular details such as service invocation metrics, i.e. how many times the service was requested, response time, service authorization, and interactions between services. For example, a business may want to track how many times a product was purchased during a particular time period, and how many times it was out of stock. Service tracking at the application level will prove to provide significant technical and business benefits.

In terms of SOA lifecycle, control is required at all phases – design, deployment and operation. As mentioned earlier in the paper, the main value propositions behind SOA will be reuse, efficient development, simplified maintenance, and portability. As SOA implementation expand, companies will need a catalog of services and an approval/workflow process, i.e. governance, to ensure that the value propositions are realized.

7 Conclusion

According to OASIS, the Service-Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among interacting software agents or services.

Services integrated into SOA come in two different types: coarse-grained and fine-grained. The granularity is based on how much of a service is exposed. The SOA model implies the loose coupling of services of optimal granularity to support the desired business processes

A primary goal of implementing Service-Oriented Architecture is to simplify development and implementation of new applications and capabilities through aggregating many low-level tasks into higher-level services.

A secondary goal of SOA is to enable any application or process to utilize the services of any other application or process, over any means of interconnection.

The universality of SOA can help enterprises use their existing systems in new ways, for new purposes, for greater competitive advantage, efficiency, and return on investment.

As today's IT challenges continue to broaden and the move to SOA increases, a fully integrated platform focused on manageability and interoperability will be the key to meeting and exceeding the challenges of cutting edge industries.

References:

[1] http://www.oasis-open.org

[2] ***. Enterprise Service Bus (ESB) Solution. http://www.capeclear.com, 2004.

[3] ***. ESB: Evolving Beyond EAI. http://www.iona.com, 04/2005.

[4] Finkelstein, Clive. The Enterprise: Business Process Management Languages, Part 1: BPEL. http://www.dmreview.com/editorial/dmreview/, 02/2005

[5] Clabby, Joe, *Web Services Explained*, Prentice Hall PTR, Upper Saddle River (NJ), 2003

[6] Newcomer, E. *Understanding Web Services*, Addison-Wesley, Boston (MA), 2002

[7] Kaye, D., *Loosely Coupled: The Missing Pieces of Web Services*, RDS Press, 2003

[8] Lefevbre, Alain. The True Nature of Web Services". http://www.intranetjournal.com/articles/200106/pap_06_13_01a.html, 13/06/2001.

[9] ***. "Principles of BPEL, Orchestration, and the ESB". http://www.capeclear.com, 2004.

[10] Taylor, James. Decision Management Applications. http://www.dmreview.com/editorial/dmreview/, 05/2005