webMethods®

Get There Faster.™

The Business Case for SOA

Rationalizing the Benefits of Service-Oriented Architecture

January 2005

TABLE OF CONTENTS

INTRODUCTION **4**

THE PROMISE OF SOA **5**

SOA CONCEPTS 5

SOA BENEFITS 6

Reuse: Accelerated Implementation, Lower Effort And Risk *6*

Composite Applications: Unique Solutions To Empower Users *7*

Loosely-Coupled: Greater Flexibility, Increased Implementation Agility *8*

ESTABLISHING AN SOA IMPLEMENTATION STRATEGY **10**

THE NEED FOR SOA INFRASTRUCTURE 11

JUSTIFYING THE INVESTMENT IN SOA **12**

CONCLUSION **13**

INTRODUCTION

One of the most significant IT initiatives underway is the adoption of service-oriented architecture (SOA), where IT assets are aligned to business services in a standard, flexible and architected fashion. The concept is often discussed in conjunction with Web services, though the two are not synonymous. In fact, SOA predates Web services by many years, but it is only now that SOA has hit the headlines. This is due in part to the fact that Web services have been a useful development that makes it much easier to deploy an SOA.

The promise of SOA is that it can transform the IT assets of a business, making it possible to do more with less, and faster. It is a key development towards the goal of a truly agile business where new business initiatives can be deployed as needed—with the necessary underlying IT support—with minimal delay. But there is also a concern in many CIOs' minds that SOA is a radical new form of software infrastructure requiring extensive changes across the entire range of IT systems. In today's business climate, getting the investment to fund such an exercise would be hard (if not impossible), even though the attractions from an IT perspective of a clean, component-based architecture are considerable.

So, how to make the business case for SOA?

In fact, the rationalization is not as hard as it might seem. First, because of the software engineering "feel" of SOA, there is a tendency to look at it as delivering purely internal IT savings. Justifying expenditure on this basis alone can be extremely difficult. However, once the more significant returns of business effectiveness and agility are taken into account, the justification looks much more positive.

Secondly, SOA does not need to be fully implemented and deployed before benefits can accrue. In fact, SOA is ideally suited to incremental deployment, where investment can be made on a step-by-step basis tied to individual business projects. Returns also start to be realized on an incremental basis, thereby removing a major element of business risk in any justification process and greatly increasing the likelihood of acceptance of the business case.

By taking these factors into account, it becomes possible to make a strong case for adopting the SOA vision and deploying it in all new projects. However, in order to avoid surprises further down the line, the business case needs to take into consideration the requirement not just to implement SOA concepts, but also to put into place an enterprise-class SOA infrastructure. It is this infrastructure that makes SOA real, makes it useful, and transforms it from being a concept into a true enabler of the agile business.

THE PROMISE OF SOA

The concept of SOA has been around for some time. Although the term first emerged in the 1990's, the spirit of SOA existed long before that. The underlying principles of SOA—componentization, encapsulation, separation of interface from implementation, loose-coupling, etc.—have been promoted since the early days of computing. However, achieving these goals was, in many ways, easier in the days of centralized computing and when the focus was on IT resource optimization.

Then the landscape changed. On the technology side, distributed computing arrived, offering great productivity and price-performance, and the Internet emerged, promising ubiquitous connectivity. On the business side, IT was seen increasingly as something that could be used for business advantage. But these changes introduced considerable challenges in linking different technologies and platforms together and mapping the IT infrastructure more closely to business need. The advent of object-oriented (OO) computing paved the way that eventually led to SOA by introducing the concept of having discrete program components with standard inputs and outputs that could be used as building blocks to assemble applications.

SOA Concepts

In SOA, these application building blocks provide a specific "service", that is, a business process step such as "Create Invoice" or "Customer Lookup". These services are implemented with standard ways to invoke them and are "loosely-coupled" —that is, they can be invoked without the caller needing to understand anything about the technology choice or location of the service provider. Thus, the "Customer Lookup" service can be invoked from any other business application that requires customer information. In the days of application silos, where monolithic program stacks dealt with particular business needs, there would have been different versions of code in every stack to retrieve customer information, but in an SOA there is only need for one.

The other elements involved in SOA are a "transport" layer to allow communications between the different services, wherever they are, and some sort of directory—a conceptual Yellow Pages—that enables services to be identified and located. In addition, for an SOA to be effective in the real world, there are numerous other "infrastructure" capabilities that are needed, such as security, versioning, auditing, management, monitoring, and predictable quality of service. In most SOA implementations, these functions would be handled by some sort of SOA infrastructure.

A related topic to SOA that should be discussed is Web services. As mentioned, in an SOA, services are described, found, and invoked in a standard way. Web services provide a standard approach for addressing these needs. With Web services, standards specifications are defined that handle how a service is described (WSDL), the invocation communications (SOAP), and the identification and location of the service (UDDI). This makes Web services ideally suited for SOA implementation (though by no means are they the only possible approach to SOA), and, today, many people talk of SOA and Web services synonymously.

SOA Benefits

In order to build a business case for SOA, it is necessary to consider the financial returns. However it is also worth considering the full range of benefits of an SOA, since there may be some benefits that are highly desirable although difficult to evaluate in monetary terms.

At first glance, there are three features of an SOA provide value. These are:

- The promotion of reuse
- The ability to combine services into new, composite applications
- The use of loosely-coupled services through a standard interface

The following sections of the white paper consider each of these areas and describe the related financial benefits or value-added soft benefits.

Reuse: Accelerated Implementation, Lower Effort And Risk

Of all the features of an SOA, reuse is the one that has the clearest range of benefits. The benefits of reuse can be broadly addressed in two major categories:

- Benefits from reusing components to reduce redundancy
- Benefits from reusing components when delivering new functionality

Firstly, as already stated, many IT implementations have evolved into a discrete set of functional silos, usually aligned with different business units. So, for example, a retail company may have a silo for the distribution systems, a silo for the stores systems and a silo for corporate functions. Typically, these silos will have been developed on different IT platforms, using different programming languages, and there will almost certainly be many instances of duplicate functionality across silos. An SOA enables companies to avoid these duplications, creating single services shared between all applications.

Avoiding duplications translates into real cost savings when changes need to be made. Suppose a company decides that customer details should now include the customer email address. In the silo scenario, every “Customer Lookup” implementation has to be changed, which requires multiple development and test efforts, and skills in all the different technology platforms.

In an SOA environment, the single version of the “Customer Lookup” service has to be changed and tested once only, using the single skills set with which the service was developed. The real money benefit is **a reduction in the cost of maintaining the application portfolio**. There is also another benefit here, although harder to evaluate in monetary terms. In the above example, the change to include customer email information can be implemented much more quickly since there is only one place where code changes are needed. This makes IT more responsive to the need for business change, and **enables the business to update its operations more quickly**.

The maintenance savings with SOA can be substantial. If a company is able to reduce the amount of maintenance required by just a few percentage points, this can work out to

be an enormous financial saving since application maintenance accounts for a significant portion of the IT budget. However, an even bigger impact comes from the second aspect of reuse, that of the ability to reuse components when delivering new operational processes.

By decomposing an application implementation into discrete business services, and allowing these services to be recombined readily, systems can be delivered that reuse existing building blocks and **reduce new development work solely to new areas of functionality**. As an example, consider a company offering home loans over the Internet. The company now decides to start offering car loans too. This needs a new IT-based service. But how much of the existing process needs to be changed? If it is divided up into discrete units through the use of an SOA, then the primary changes will be around the insurance details and validation; other aspects of the solution can simply be reused.

This example illustrates how powerful reuse is in the delivery of new business functionality. Because much of the new service reuses existing components, the new development work is minimized. This means:

- Companies are able to implement new processes much more quickly
- The development and test cost is reduced significantly
- The business risk of an implement defect or intermittent service is reduced

The last point may not be immediately obvious, but the point is that by using existing components (that are known to work properly) wherever possible, the risk of injecting errors is limited to the areas where there is new development. Therefore the risk of software problems disrupting the business are minimized, enhancing quality of service, and **reducing business risk**.

The **reduction of development and test costs, and rework** due to defects represent clear financial benefits for SOA. But the ability to deploy new services to support operational changes in the business more quickly can result in **improved business effectiveness** through increased agility and **faster competitive response**. If these factors can be assessed as delivering increased market share, for instance, then the financial benefit can be huge.

Composite Applications: Unique Solutions To Empower Users

In the same vein as the previous example, SOA makes it feasible to assemble an entirely new class of business application that combines functionality from multiple existing systems to provide end-users with new cross-functional capabilities.

An example might be the provision of a self-service portal for consumers or business partners. Here, a number of existing processes supported by IT services might be combined within the portal to provide the end-user with a one-stop access to a range of company products. This type of composition would have been difficult without an SOA approach, requiring considerable application integration, programming and testing effort. But with an SOA, the “composite application” can be assembled easily, regardless of the

location or technology choice of the different services being drawn together. This allows IT to **respond with greater speed to business demands**, keeps costs to a minimum, and powers imaginative new ways to address the needs of end-users more effectively.

Loosely-Coupled: Greater Flexibility, Increased Implementation Agility

The next area to consider is the "loosely-coupled" characteristic of SOA, where the invoker of a service does not need to be aware of its location or technology platform. This introduces a high degree of flexibility of choice, in that decisions about a particular service component—in terms of what platform and technologies it uses and where it will run—are unconnected to any other user of the component. This provides several benefits.

First, it may be that a particular service would be best suited to running on a specific hardware platform or in a specific geographic location. Computing-intensive business algorithms, for example, could be deployed where price/performance is optimal for this type of activity. Other variations might be to ensure that network traffic is optimized or that a particular service remains available in the case of network or system failures. This flexibility to run things where it makes the most sense can **increase IT efficiency, improve quality of service,** and **reduce costs**.

Second, because the invocation interface to services in an SOA take a standard form, it removes the need for a programmer creating a new service to understand the technologies used by individual services. This means a reduction in skills requirements across the development organization, leading again to **cost reductions**.

Another aspect of this flexibility of choice is that services can be sourced in the most efficient and effective fashion. One service might be written internally, another could be replaced by a package, while yet another might be sourced from an external supplier. Credit checking is a typical candidate for 3rd-party sourcing, for instance. The result is that the most efficient and business-effective option can be chosen in each instance. Indeed, with SOA, the sourcing of a service can be changed without disruption to the overlying system.

Third, in the case of connectivity with business partners, the loosely-coupled, standard interfaced services approach provides additional benefits. Using an SOA, it becomes relatively easy to offer a range of services externally for a partner to use. Because of the transparency of location and technology offered by an SOA, the partner does not need to be aware of how the service is implemented, and because of the standard interface approach the partner needs only to have a minimal amount of information to invoke the service. And if the partner has also adopted an SOA approach, the reverse is true where a partner-supplied service can be hooked into an internal business operation quickly and easily. This aspect of SOAs promises **improved process efficiency** and a **higher degree of automation across the enterprise value chain**.

Finally, another by-product of the loosely-coupled facility of an SOA is the ability deploy incrementally. This is because, as already discussed, loosely-coupled components can be invoked without the caller having any knowledge of how they are implemented. The

only requirement is that they can be invoked through a standard interface. Therefore, by providing wrappers for in-place applications that respond to the standard invocation interface and then mapping this to the application's native call interface, the in-house component now becomes an active SOA building block. For example, a number of vendors now offer technology to wrapper legacy applications to make them look from the outside like Web services. These can now be invoked in an SOA just as any other Web service.

The implication is that in-place investments can be leveraged in the new SOA non-invasively. Thus an SOA can be deployed step-by-step, with new services replacing legacy applications as they are updated to implement the SOA concepts natively. This not only **reduces business risk** but also **reduces payback time** on an SOA investment by allowing benefits to be realized as each component is brought into the SOA domain, rather than requiring massive investment before any benefits can be delivered.

In summary, it becomes apparent that in business terms, the benefits of an SOA fall into three areas:

- **Business Effectiveness**
  - Agility, responsiveness to market and competitive dynamics
  - Greater process efficiencies
  - Deployment of resources based on business needs
- **Cost Efficiency**
  - Reduced maintenance costs
  - Reduced skills and effort to support business change
  - Price/performance optimization based on freedom to select platform, technology, and location independently
- **Reduced Risk**
  - Higher level of IT quality
  - Incremental deployment
  - Improved payback times

These areas should form the nucleus of any business case for SOA deployment.

ESTABLISHING AN SOA IMPLEMENTATION STRATEGY

One approach to adopting SOA might be to focus on the creation and utilization of the actual services. For example, a company might decide to standardize on Web Services as its service delivery vehicle, and spend its time developing wrappers for existing applications and writing new Web services to deliver the SOA benefits. While the numbers of services are small, this approach may be feasible. It requires minimal planning, preparation and investment, and is often the default choice made by companies as they start to dabble with Web services.

However, as the number of services increases—keeping in mind that the real benefits of SOA accrue with widespread usage across the enterprise—the deficiencies of *ad hoc* deployment quickly become apparent. These include:

- Inability to properly manage and monitor services
- Inadequate security measures
- Inconsistent performance and reliability (quality of service)
- Issues resulting from a lack of discipline and control in the development and operations management areas
- The need for middleware functionality to handle orchestration, transformation etc

Security is a problem even in the early stages of the life of an SOA. The challenge is that SOAs are designed to make it easy to access particular business services. This ease of access—while tremendously beneficial for other reasons—is a pitfall from a security perspective, so it is vital that some sort of mechanism is provided in an SOA to police access to services.

Another important point to recognize is that a range of additional, middleware functions are usually required to make SOA useful for delivering business solutions. For instance, although SOA addresses the issues of how to invoke and run a particular service, it does not deal with the situation where data is expected in different formats by different services. Hence some sort of transformation capability will be needed. The ability to "orchestrate" services is also fundamental, to link services together into the right sequence (taking into account the necessary rules and conditions) to support a particular business process.

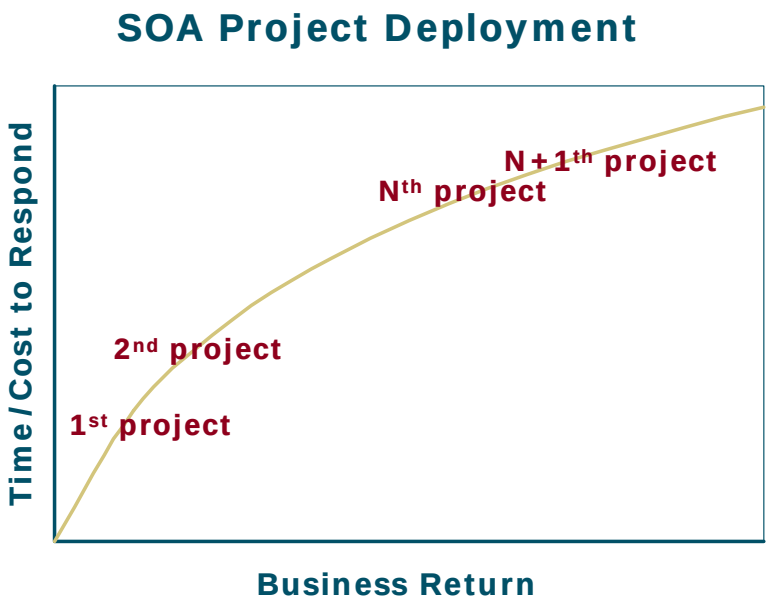
Once an SOA is deployed in production, even at fairly low levels of usage, there will be a need to monitor and manage the operations of the SOA. How will new services be added, or existing ones versioned? How will currently available services be modified dynamically? In the event of poor performance, which is the service causing the problem? All these questions reflect classic management and monitoring issues, and an SOA needs to address them.

What quickly becomes clear is that neither SOA or Web services, on their own, fulfill these functions. Rather, there is **the need for an SOA infrastructure that exists external to the services** to address the needs listed above.

The Need for SOA Infrastructure

Deploying an SOA infrastructure solution as part of the overall SOA implementation strategy is vital if companies are to achieve a scalable, reliable, environment that delivers the benefits that SOA promises. Although it might be possible to address the issues above to some extent on a tactical basis in the early stages of an SOA implementation, the desire for greater business returns will force the expansion of the SOA beyond any level that could be handled on this basis.

For business returns to continue accruing and to accelerate with successive projects, the SOA environment has to be scalable. A key requirement for achieving scalability is ensuring that the complexity of the environment is managed. As the graph below indicates, the benefit of a managed environment is that projects become faster to implement—and business returns increase—with each successive project. (Time to implement is greater, and returns are less, for initial projects because there is a smaller inventory of existing services to reuse).



As mentioned, an SOA infrastructure is a product that exists independently of the services themselves to deliver capabilities such as security, versioning, load balancing, and so on.

Fortunately there are commercially available offerings to satisfy the SOA infrastructure requirement. Given the incremental deployment approach that is one attraction of an SOA, these offerings typically can be deployed in stages so that not all the costs of implementation have to be borne up front.

JUSTIFYING THE INVESTMENT IN SOA

To build the business case for SOA, a number of points should be taken into account:

- An SOA can be deployed incrementally, allowing returns to flow more quickly
- As the SOA becomes more widespread, business value increases rapidly
  - Implementation costs per project fall
  - Maintenance costs drop
  - Business effectiveness improves
  - Time to market for new business initiatives shortens
- To provide the level of service the business needs, a managed infrastructure is required, but the cost can be amortized across multiple projects

The cost efficiency benefits can usually be expressed in a monetary form or can be assessed in such terms fairly easily. The business effectiveness benefits—improved responsiveness, reduced implementation risk—are more difficult to evaluate but generally more important. In order to evaluate these benefits, and the ones about business risk, the IT team should liaise with the line of business to get credible value assessments.

On the cost side, a number of areas should be considered, including:

- SOA architecture, design, and planning, i.e., the upfront effort required to define the overall SOA model to be adopted, along with the technology selection
- Organizational implementation to support SOA, e.g., the implementation of an oversight committee and a competency center to manage the deployment
- Education and training costs
- License and maintenance costs for an SOA infrastructure solution
- Additional development costs for each project as required

The result of this analysis should be a model that clearly demonstrates the business and IT value that SOA can deliver.

CONCLUSION

Service-oriented architectures are not a passing fancy, and have not suddenly emerged from nowhere. The SOA concept is a logical extension of IT theory and concepts that have developed steadily over time. But the value of an SOA should not be underrated. It brings the well-known concept of reuse to software implementations—a concept that has been proven again and again to contribute major business value in other industries such as manufacturing.

The company that successfully deploys a wide-spread SOA will find itself able to respond to market opportunities and competitive threats with a speed and agility that is not possible for companies still constrained by more traditional IT implementations. The real challenge is how quickly and adeptly organizations are able to mobilize themselves to take advantage of this change.

ABOUT WEBMETHODS, INC.

webMethods is a leading provider of business integration software to the Global 2000 and large government agencies. Our technology lets our customers integrate, assemble and optimize available IT assets to drive business process productivity. Currently, more than 1,200 customers are meeting customer demands, reducing costs, creating new revenue opportunities and reclaiming ROI. Faster. webMethods is headquartered in Fairfax, Va., with offices throughout the U.S., Europe, Asia Pacific and Japan. More information about the company can be found at www.webMethods.com. NASDAQ: WEBM.

Worldwide Headquarters
3930 Pender Drive
Fairfax, VA 22030
USA
Tel: 703 460 2500
Fax: 703 460 2599

US West Coast
432 Lakeside Drive
Sunnyvale, CA 94088
Tel: 408 962 5000
Fax: 408 962 5329

European Headquarters
Barons Court
20 The Avenue
Egham, Surrey
TW20 9AU
United Kingdom
Tel: 44 0 1784 221700
Fax: 44 0 1784 221701

Asia-Pacific Headquarters
Level 15
Philips Building
15 Blue Street
North Sydney NSW 2060
Australia
Tel: 61 2 8919 1111
Fax: 61 2 8920 2917

webMethods Japan KK
Izumi Garden Tower 30F
1-6-1 Roppongi, Minato-ku
Tokyo 106-6030
Japan
Tel: 81 3 6229 3700
Fax: 81 3 6229 3701

www.webMethods.com