A Pragmatic, Business-Focused Approach to SOA
Contact Information

Milan
Enrico Strada
Partner
+39-02-72-50-93-00
enrico.strada@booz.com

Paolo Lasagni
Principal
+39-02-72-50-93-01
paolo.lasagni@booz.com

Munich
Dr. Tim Habermann
Principal
+49-89-54525-679
tim.habermann@booz.com

Andre Gajda also contributed to this Perspective paper.
EXECUTIVE SUMMARY

Judging by numerous companies’ satisfaction with service-oriented architecture (SOA), its benefits are real. SOA’s powerful combination of technical flexibility, standardization, and openness can help solve the many technical problems companies face as they move beyond proprietary systems and toward modular, reusable services that can make business processes more agile.

But to business users, the cost of implementing SOA can be dauntingly high, and the business benefits remain elusive. The very phrase service-oriented architecture means little or nothing to most of the executives who are ultimately responsible for green-lighting and funding large IT architecture projects. SOA looks to them like just one more in a stream of IT abbreviations that rarely deliver all their overhyped business benefits.

Even among IT practitioners, misconceptions continue to swirl around the definition of SOA and what it really delivers in terms of business benefits, causing further ambiguity in their attempts to explain the value of SOA to business users. The result is significant confusion about the technology. By providing the business side with sufficient reason to avoid the high costs and risks involved in moving to a full-scale SOA solution, this confusion has significantly impeded SOA’s adoption at many corporations.

We believe that the technological underpinnings of SOA have matured to the point where many businesses...
should begin adopting SOA design principles—at least for some parts of their IT architecture—to improve their competitiveness and agility. In our work with numerous companies, however, we often observe that promising SOA initiatives do not get approved because their benefits cannot be made transparent to the business. And we sometimes come across instances in which SOA is deployed inappropriately, leading to increased implementation complexity with very limited business benefits.

To avoid such missteps, IT professionals must take a pragmatic, business-oriented approach to explaining the virtues of SOA, one that clearly and carefully balances the project risks and business benefits of the new architecture. This Perspective paper aims first to give IT professionals a clear explanation of the fundamental concepts of SOA and its business benefits, and then to put forth an effective approach for addressing the technical and business challenges inherent in implementing an SOA program.

Key Lessons

Focus on business pain points when selecting projects for SOA. This will determine where they are most needed and help to sell SOA to the business.

Understand—and make business users understand—that the payoff from SOA will come over time.

Chose the scope of the project appropriately. It should usually be larger than one legacy application but not necessarily as large as an entire IT domain—and never the entire enterprise!

Reuse legacy applications where appropriate, rather than trying to do everything in one shot.

Make data normalization a priority; SOA will not deliver much if the data model remains inconsistent.

Avoid making significant investments in SOA-enabling technologies, since many ERP and EAI packages from the leading vendors already support SOA technologies.

Install proper IT governance structures to manage service inventories across traditional application and domain boundaries.

Often, promising SOA initiatives do not get approved because their benefits cannot be made transparent to the business.
**CLARIFYING THE TERMINOLOGY**

What exactly is service-oriented architecture? Too often, the phrase means different things to different people, or it’s simply misunderstood—not just by business users, but by IT professionals. It is not one specific technology or a specific IT standard. Nor is it an IT product or an application framework. Any claim that a product is “SOA compliant” can be ignored as pure marketing hype.

Rather, the phrase service-oriented architecture refers to a design paradigm for IT architectures that is aimed at separating business processes from the technological implementations—the programming languages and application frameworks—intended to support them. That decoupling is the key technical benefit of SOA. The result is a high degree of modularization, as well as the ability to reconfigure applications quickly and with reduced integration effort. These benefits lead directly to the overarching business benefits of SOA: closer alignment of IT with the needs of the business and increased agility.

SOA is based on a hierarchy of “services”—defined as applications that implement a certain amount of business logic, such as a credit check for a potential new customer or the retrieval of a customer record. These services connect with each other through standardized interfaces. The hierarchy of services can be rearranged quickly in order to reflect changes in business processes. Typically, SOA implementations consist of three classes of services:

**Utility services.** These represent enterprise resources, such as a command to print and send out a document or archive a document as a PDF file. Utility services are typically rather generic and can be used in many applications and among multiple users throughout an enterprise, often beyond the boundaries of a single IT domain.

**Entity services.** Entities correspond to real-life objects in the enterprise data model, such as customers, invoices, or inventories. Entity services typically implement critical fundamental processes for these real-life objects, such as “change customer address,” “send out invoice,” or “process payment.”

**Task services.** These services embody more complex business logic—such as processing an order and sending out the product, or automatically managing an inventory—and make use of utility and entity services as well as other task services. Whereas utility and entity services typically remain relatively stable, task services are generated and modified more dynamically to reflect changes in business processes.

Among the many virtues of SOA is that once the necessary services are created, they can be reconfigured and reused in any number of different contexts. But that will work only if the data involved is consistent and works within a standardized interface with all the services that make up the overall system. Consistent data and standardized interfaces allow applications to be connected without the usual complicated and time-consuming process of integrating applications, which involves developing point-to-point interfaces among applications and converting the data at each point. The simplicity of reconfiguration potentially offered by SOA applies not only within individual companies but also across company boundaries to suppliers and business partners.

Thus, the success of an SOA implementation depends on developing a consistent data model and a standardized interface. SOA is usually, but not always, implemented through the use of Web services technology with
defined standardized interface descriptions, called service contracts. Web services can be developed in practically any technology or programming language as long as they support the standardized interfaces. Developers can integrate legacy applications into an SOA architecture by building “wrappers” around their application programming interfaces that comply with the Web services standards. An integration infrastructure ensures that Web services used to create each service can interact with each other (see Exhibit 1).

Once a collection of Web services that use a common data model and standardized interfaces has been created, the collection can be used to implement more complex business logic or entire business processes. Advanced applications, known as business process management tools, allow the graphical management of business processes that are based on existing elements of business logic and implemented in Web services. Once a company has developed a repository of services with critical mass, the implementation and modification of business processes can be significantly simplified and accelerated. That is where SOA can really improve the alignment of IT with a company’s business processes.

Standard interfaces and the Internet make up SOA’s core communications platform, streamlining the integration of services across company boundaries and making outside entities an integral part of a company’s IT architecture. The reduction of integration effort, both internally and externally, will over time also lead to shifts in the value chain of application development. Today, it is normal for system integration to consume a large share of every development project (even with integrated systems such as ERP platforms). SOA, on the other hand, will allow companies to significantly reduce the share of IT project budgets devoted to integration. At the same time, standardized interfaces allow for the flexible exchange of application components in the IT architecture and thereby reduce companies’ dependence on certain vendors or system integration partners. This again will drive IT costs down in the long run.

Despite its technical complexity, SOA really shouldn’t be an especially difficult concept for business users—or the executives who must give the go-ahead for any SOA implementation. Put simply, SOA involves the creation of small IT “services” that can be used, reused, and redesigned for automated business processes when needed. And SOA can accommodate old, inflexible legacy and ERP systems through the development of specific services that can use the data held in those systems.

Still, actually implementing SOA can be a major challenge for any company, and especially those that do not adhere religiously to the business reasons for implementing it in the first place. With that in mind, we turn to the practicalities of implementation.

---

**Exhibit 1**

**SOA Layers**

- **Data Layer**: Databases store data items for legacy applications and Web services.
- **Application Integration Layer**: SOA wrappers access functionality in legacy applications Web services-enabled applications allow the SOA platform to access their functionality directly.
- **Hierarchy of SOA Services**: Functionality from underlying applications is grouped into a hierarchy of utility, entity, and task services.
- **Business Services**: SOA services are used to model business processes or to build composite applications.

Source: Booz & Company
A PRACTICAL APPROACH TO IMPLEMENTING SOA

A Focus on Business Pain Points
The implementation of service-oriented architecture must never be an end in itself. We do not recommend launching SOA projects simply in order to re-create existing systems in a service-oriented way. SOA should be used as the architecture paradigm in IT projects that have either a clear business impact through the creation of new capabilities or an IT impact beyond service orientation, such as the replacement of older, no-longer-viable systems.

In order for an SOA implementation to show early successes, and thus convince decision makers on the business side of its benefits, it needs to demonstrate clear and tangible advantages. So it is critical to select an appropriate IT domain for that implementation. SOA is especially powerful in domains where packaged IT solutions do not exist or do not deliver the required process flexibility; an example would be a system architecture for activation and provisioning of services in the telecommunications industry, where new services are frequently developed and need to be integrated quickly. We recommend that companies begin implementing SOA in domains with a strong demand for flexibility in business processes that has not previously been met with the use of packaged solutions. If a company has deployed a packaged, integrated ERP solution that, for the most part, adequately supports the business processes, there is little to be gained by replacing that ERP package with a set of newly designed services. On the other hand, in heterogeneous systems that have grown on an ad hoc basis over several years, SOA can help considerably in removing the technological barriers to business-driven innovation, such as high complexity, high costs, and slow time-to-market.

Case in point: The U.S. branch of a large European investment bank was looking for ways to remove the constraints that its legacy applications were putting on the automation of its financial transaction processes—constraints that were ultimately inhibiting business innovation. After studying its options, the bank’s IT department decided that implementing a set of business process management tools embedded in SOA would be the best and cheapest way to further automate its legacy trade execution applications. The services were configured to have access to the necessary legacy applications, with the result that the transaction processes could be fully automated. This in turn allowed for the “straight-through” processing of financial transactions.
without human interaction, enabling the bank to reduce the number of staff needed to process transactions, to increase real-time throughput, and to eliminate manual processing errors. Moreover, the IT department could then reuse more than 40 percent of the newly created, highly standardized SOA services to automate other business processes, such as customer service, which decreased time-to-market for new financial services. And the SOA project cost turned out to be a fraction of the cost of the alternative: upgrading the bank’s legacy applications.

It is just that kind of clear, powerful business rationale that IT departments must develop if they hope to get approval for an SOA implementation. In discussing SOA, we intentionally avoid the term “business case,” as it suggests the calculation of a distinct net present value analysis for the implementation. We rarely see the business case for SOA truly meet its goals in individual IT projects: The real value of SOA comes only over time, as companies continually reuse and reconfigure the services they initially built (see Exhibit 2). Eventually, that effort pays off in increased agility and greater alignment between IT and the business. IT proponents can make the argument for investing in SOA much more powerfully if they concentrate on its ability to affordably meet critical nonfinancial business goals, such as reduced time-to-market and better data consistency.

*The Scope of an SOA Deployment*

IT organizations should use caution when defining the scope of the domain for an SOA implementation. It should be large enough to allow for the reuse of the services being created; a single existing integrated application may be too small to reap the full benefits of reuse. On the other hand, it should not be so large as to become unmanageable. Too broad a scope runs the risk of failing either technologically or commercially. Once the project is far enough along to be considered a success, the scope of the SOA implementation can begin to be extended.

The happy results of a properly defined domain can be seen in the case of a large high-end British retailer that faced problems with inconsistent information it had gathered about customers and their shopping patterns. Information about individual customers was spread across several databases and could not be accessed in a consistent way from a single point. The problem was significantly affecting the company’s competitiveness in its lucrative high-end segment, and company leaders concluded they needed to put a solu-

---

*Exhibit 2*

SOA Investment Cycle

Source: Booz & Company
tion in place in just four months. To achieve the results, the company’s IT department had agreed with the business side to use SOA to integrate and standardize the customer information, and agreed that the implementation would be strictly limited to those applications and databases that processed customer information.

Because the scope of the project had been so clearly defined, the company’s IT department succeeded in completing the project within the four-month time frame. Standardized customer information allowed the company to create a more effective customer loyalty capability, which the business side saw as a tangible business benefit. The services created in the course of the project were ready to be reused in other projects involving such areas as customer care. And in the long term, the first SOA project’s success increased the willingness of the company’s top management to extend SOA to its remaining legacy applications.

The Sequence of Implementation Steps
Once SOA has been approved as the design paradigm for a particular project within a domain, its implementation requires a more fundamental approach than traditional application development does—it involves an overhaul of the entire architecture rather than development of a single application within the existing architecture. A successful SOA deployment depends on a common and consistent data model within the domain and the clear allocation of the applications maintaining the master data. Consistent data is an important prerequisite for the definition of powerful entity services and for service normalization—the avoidance of any functional overlap between services. Entity services can represent business objects consistently only if the data within the domain covered by the project is harmonized and normalized. Thus, every SOA project requires that the data be normalized at the beginning. Early on, the data can be normalized on a conceptual level alone, but ultimately physical redundancy should be reduced as much as possible.

Based on the normalized data model, an inventory of services needed can be designed, still on a conceptual level. The services are orchestrated in a hierarchy. As a concept, SOA does not demand that services cover a specific scope; rather, the proper level of modularity should be chosen pragmatically. Designers should keep in mind that higher modularity makes services more reusable, but that must be balanced with higher modularity’s propensity to make design more complex and to negatively affect performance owing to communication overhead. If the IT department and the business are hoping to extend the scope of the SOA domain at a later stage, it is important to take into account the needs of adjacent IT domains when designing the services.

Once the hierarchy of services is designed on the conceptual level, the services’ mode of implementation can be defined. This can range from wrapping existing applications and databases in an SOA-enabled interface, typically in the form of Web services, to coding new applications from scratch using a state-of-the-art programming language. Here again, we recommend that the amount of new code be kept to a minimum, and that existing applications be used as much as possible. Because existing applications are known quantities, with proven functionality and performance, keeping them in place reduces a project’s technical risk. If the wraparound services are designed properly, it’s simple to replace legacy applications with new services later on, as the hard work of integrating the wraparound services with the new ones has already been done.

Once the new architecture has been designed, it’s time to implement, test,
and deploy the new SOA services. These days, most large software vendors, especially in the EAI and ERP area, provide their own SOA-enabling products—SAP offers NetWeaver, Oracle has its SOA Suite, and IBM promotes its WebSphere—which are already well integrated with the vendor’s other software offerings. Thus, if a company is already using a vendor’s applications, it can lower the barrier to adopting SOA, protect its previous investments, and accelerate the implementation process by using the SOA components offered by that same vendor. Should the technical capabilities of one vendor’s platform not be adequate to meet a company’s long-term needs, especially in very heterogeneous environments, then the overall design as well as the individual services can be migrated to another vendor’s SOA platform, which should be selected according to how well it meets those future needs.

**Organizational Changes**

Taking full advantage of SOA isn’t just a matter of designing and implementing the SOA services; the processes affected, and even the organization itself, must also be adapted. Because many of the newly created services will eventually cross the boundaries of various domains, those maintaining the inventory of services must rethink the IT governance structure in order to develop a cross-domain perspective. Without that, it is unlikely that a company will succeed in fully leveraging the potential of SOA. If the service inventory is managed only within organizational boundaries, siloed applications will just be replaced by siloed service inventories. This will inevitably lead to duplicated and inconsistent implementation of services that limit SOA’s potential to reduce the effort involved in implementing cross-domain applications (see Exhibit 3).

---

**Exhibit 3**

**SOA Demands a Cross-Domain Approach to Governance**

<table>
<thead>
<tr>
<th>TRADITIONAL IT GOVERNANCE</th>
<th>IT GOVERNANCE IN AN SOA CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain A</td>
<td>IT Strategy</td>
</tr>
<tr>
<td>Domain B</td>
<td>Service Inventory</td>
</tr>
<tr>
<td>Domain C</td>
<td>Data Model</td>
</tr>
</tbody>
</table>

Source: Booz & Company
CONCLUSION

When broken down and described in plain English, the fundamental concepts of SOA need not be all that hard to understand, even for business users. Similarly, a pragmatic approach to identifying the kinds of projects suitable for SOA can help business users understand SOA’s potential value within a limited context. Finally, when seen as a series of clear steps, the process of implementing SOA will look a lot less daunting to business users. We are confident that this approach to SOA will not only make it more attractive to business users, but also allow it to be successfully deployed, and thus enable SOA to live up to its hype.

Endnotes

1 Technically, SOA can also be implemented without Web services technology. However, in practice, Web services technology has become the standard basis for SOA.

2 In the terminology of SOA, the concept of modularity is referred to as “granularity,” i.e., the smaller and more modular a specific service, the higher its granularity.
Booz & Company is a leading global management consulting firm, helping the world’s top businesses, governments, and organizations.

Our founder, Edwin Booz, defined the profession when he established the first management consulting firm in 1914.

Today, with more than 3,300 people in 58 offices around the world, we bring foresight and knowledge, deep functional expertise, and a practical approach to building capabilities and delivering real impact. We work closely with our clients to create and deliver essential advantage.

For our management magazine *strategy+business*, visit www.strategy-business.com.

Visit www.booz.com to learn more about Booz & Company.