

The SOA Magazine

Feature Article



SOA Engineering Misconceptions

by Ted Barbusinski

Published: March 13, 2008 (SOA Magazine Issue XVI: March 2008)

[Download this article as a PDF document.](#)

Abstract: We have all been saturated with information from credible sources underscoring the risks, as well as the potential for ROI, associated with strategic IT SOA investments. In a recent press release [REF-1], Gartner noted that "bad technical implementations", as well as a lack of governance, are key contributors to the risk of failure in strategic SOA initiatives. Although the issue of governance is frequently addressed by various sources, the causes of "bad technical implementations", and the foundations for SOA technical excellence, are often misunderstood.

This is the first of a two articles that intend to offer some insights into evaluating and adjusting the technical direction of an SOA investment with an eye toward technical excellence and enduring ROI. Within this first part, we establish a basic understanding of SOA engineering goals and provide a frank overview of common misconceptions that can cause SOA technical implementations to become problematic. Part two of this article subsequently examines SOA engineering focal points and success factors that are foundational to successful SOA technical implementations.

Introduction: Understanding SOA Engineering Goals

Before we can design for "increased ROI", we need to understand what that means relative to SOA technical foundations. In a nutshell, increased ROI describes an IT enterprise that could rapidly deliver new, cost-effective business value in response to highly dynamic business requirements, while significantly reducing operating costs. At the heart of high SOA ROI are the complementary concepts of "business agility" with "operational efficiency". To understand what these concepts mean relative to SOA technical foundations, we need to decompose them into a tangible set of IT enterprise technical capabilities.

There are good reasons as to why Gartner noted that "bad technical implementations" are a frequent cause of failure. Buried within this lengthy list of capabilities are a number of subtle land mines and pitfalls that are common causes of failure in SOA technical implementations. These pitfalls are fueled by a series of common SOA-related misconceptions and cloudy perspectives that often cause even the most experienced IT professionals to overlook issues critical to a successful SOA implementation.

Armed with a cursory understanding of the technical capabilities required by successful SOA implementations, we can begin to appreciate the depth of the architectural challenge. SOA engineering is not a stand-alone, isolated endeavor. Success requires understanding and accommodating the many relationships, interactions, and dependencies between SOA solution elements and key enterprise IT assets.

From this understanding of goals and challenges, we next

Common Technical Capabilities of SOA

The following list [REF-2] associates the concepts of "business agility" and "operational efficiency" with some of the SOA-related IT technical capabilities necessary to implement and sustain them. One quick glance at this list underscores the reality that successful SOA implementations are architecturally intensive endeavors.

- Reusable Process, Information and Utility Services - Define, capture and expose core IT business process functionality and business information as easily reusable services.

examine common misconceptions that are often contributing factors leading to unsuccessful SOA solution architectures.

It's important to note that the end-to-end, multi-faceted nature of SOA engineering makes it highly susceptible to engineering problems. A problem introduced within any segment of the overall solution can result in poor ROI for the overall investment. Additionally, SOA engineering problems typically are not "soft" problems easily corrected through configuration or code changes. They tend to be structural in nature, requiring significant effort and cost to remedy.

With an eye toward avoiding problems before they happen, the discussion that follows looks at common, key SOA architectural misconceptions that can lead to serious problems (and, ultimately, jeopardize the success of the SOA implementation). Each misconception is stated, defined and then examined from the perspective of real-world SOA engineering issues.

Misconception #1: "SOA Vendor Stacks are Service Architectures"

I recently had a client (who was a key decision maker on an SOA project) ask me the question .. "What exactly is an architect?". After a moment (or maybe two) of dismay, I realized what an excellent job SOA product vendors are doing in marketing their SOA solution stacks. From the perspective of potential clients, a vendor product stack may very well constitute a comprehensive service architecture (the stack is the SOA). Why then would you need an architect?

This "SOA-in-a-box" mindset is wrong on many levels and inevitably will yield bad implementation and low ROI on strategic SOA investments. To begin with, SOA is first and foremost about enabling interoperability that can span and impact virtually all facets of an IT enterprise. How and why services are used to enable inter-connectivity between existing and future applications and resources forms much of the essence of SOA design and heavily impacts its success. Consider that, at one point or another, your service architecture and infrastructure will likely touch upon some or all of the following IT assets:

- security architecture
- portal infrastructure
- composite applications and workflows
- business processes
- information architecture
- network infrastructure
- back-end information systems
- B2B infrastructure
- mainframe platforms and applications
- data centers

- Adaptive/Flexible Business Process and Business Information Modeling - Easily (and graphically) define and/or refactor underlying business process service models and data service models in response to rapidly evolving business requirements.
- Process and Data Model Abstraction - Abstract service consumers from instabilities and dynamics within underlying process and data models.
- Service-Based Presentation - Capture common user interface-based, interactive business functionality (and associated look-and-feel) into reusable presentation components (portlet services) consumable by any standards-based business application or workflow (including the ability to support dynamic client-side presentation logic with secure XML presentation services).
- Service-Enabled Enterprise Application Integration (EAI) - Assemble complex business services that span and connect multiple systems and information sources (in other words, establish a clean fusion between EAI capabilities and service development and exposure).
- Service-Enabled B2B - Abstract common business partner interactions as reusable client and / or partner facing services.
- Composite Services - Rapidly assemble and leverage higher level business services (composite services) comprised of a set of lower-level, standalone services (component services).
- Service-Enabled Business Process Workflows - Rapidly define and assemble complex business process workflows comprised of business process, business information and presentation services.
- Service Compositions - Rapidly assemble powerful new business applications composed of reusable services woven together with a minimum of application specific "glue" logic.
- Adaptive Presentation - Host service compositions in a highly adaptive presentation framework capable of consuming presentation services in an industry standard manner (including the ability to aggregate existing business applications into this presentation framework).
- High Performance Service Interaction - The ability to efficiently consume composite services from within composite applications and workflows in a manner that yields highly responsive business functionality to end-users.
- High Availability Services - The ability to ensure mission critical services are available 24/7 and always yield high performance.

- EAI infrastructure
- COTS application integration
- IT operations systems/processes
- content management systems

Within each of these lies further sub-categories of integration issues. Strategic SOA initiatives are some of the most architecturally intensive endeavors undertaken by an enterprise. Understanding this early in the game is an important success factor.

Misconception #2: "SOA Vendor Stacks are the Best Foundation for SOA Engineering"

There is a common opinion that SOA vendor stacks are immutable, monolithic entities that embody all the elements necessary for a successful SOA implementation. From this perspective, vendor stacks are unbreakable and SOA implementations should not need to look beyond the offerings of a single selected vendor.

The attainment of long-term ROI often suffers when vendor stacks are perceived in this manner. In reality, most are built through acquisition and are integrated through open standards. Some are more deeply integrated than others. The assumption that any single vendor stack has all the best-of-breed SOA components available in the industry today introduces unnecessary and debilitating limitations into SOA engineering initiatives.

The quality and nature of the vendor products selected has an enormous impact on the technical success of an SOA project. Therefore, it rarely makes sense to address the unique, complex, interrelated architectural SOA challenges of any specific IT enterprise landscape by simply buying a monolithic stack of products from a single vendor. Highly successful SOA implementations typically utilize a limited blend of best-of-breed components from a small mix of vendors. The product selection criteria should be based on the unique challenges of the IT enterprise landscape, the strategic business goals of the SOA investment and the merits of the products themselves. Success in selecting the best SOA solution components for a specific IT enterprise requires a deep understanding of the following four problem domains:

- The IT Enterprise Landscape and Roadmap (the target IT landscape, integration challenges, architectural issues and ultimate business goals)
- SOA Vendor Community Product Offerings (especially candidate products for service-oriented solutions)
- Architectural Issues Relative to Each Component (architectural and integration issues associated with each potential product)

- Common Enterprise Service Composition and Development Frameworks - Move IT toward standardizing on a small set of common, approved service and service composition development foundations and frameworks that build toward a common knowledge base.
- Common Enterprise Service Interaction (Consumption) - Present a common service discovery and service interaction (consumption) model across multiple, heterogeneous service client platforms and applications.
- COTS Application Integration - Securely access services hosted on COTS application platforms (e.g.: ERP services, etc) plus the ability to extract and access encapsulated business functionality within vertical COTS applications (e.g.: mainframe resident applications, etc) and expose this functionality as secure process, data and presentation services.
- Service Metadata Availability / Accuracy - Ensure service interaction metadata is easily available and always accurate and authoritative.
- Service-Enabled Legislative Compliance Infrastructure - Cleanly integrate much of IT legislative compliance monitoring (e.g.: SOX, HIPPA, etc) into the SOA auditing and reporting framework.
- Service Business Activity Monitoring (BAM) and Service Metrics - Monitor key performance indicators (KPI) for both business activity and service performance and throughput at runtime enabling business reporting and SOA governance decision support.
- Central Service Governance Infrastructure and Authority - Cleanly integrate SOA governance monitoring and compliance infrastructure within SOA runtime infrastructure without degrading performance, and the ability to detect, monitor and control service deployment and govern service lifecycles, as well as impose and enforce security policies on any newly deployed service, and enforce standards compliance on all service implementations (and ensure service development consistently generates highly reusable services).
- Service Architecture/Infrastructure Scalability - Scale service functionality and infrastructure to accommodate traffic volume, WAN and geographically dispersed clients, business partners, and scale the service datacenter infrastructure, administration, and operations, as required.

Here's a list of commonly required SOA and enterprise security-related capabilities:

- Expose and consume sensitive, internally and externally facing services in a manner that ensures authorized access only.

- Architectural Relationships Between Components and IT Assets (how potential products interact with, depend upon, and integrate with each other and the target deployment environment)

Misconception #3: "An ESB Product Equals Service Infrastructure"

Another common perception that clients often develop is that an enterprise service bus (ESB) is SOA infrastructure in a box. This again is a problematic assumption because, in real life, ESB products typically focus on a specific set of infrastructure concerns often referred to as "service mediation".

The functionality typically attributed to "service mediation" features can be summarized as follows:

- service virtualization
- service security
- protocol translation
- message transformation
- context/content based service routing
- auditing and reporting
- service call error/exception handling
- service level agreements (SLA)

Since the SOA vendor community does not fully agree on what functionality should or should not belong in an ESB, some might argue that other features also be considered part of the service mediation realm (e.g.: service orchestration and BPM functionality).

Effective service infrastructure requires addressing a wide range of requirements, many of which go beyond the features provided by ESB products.

Here's a partial list of some of these additional challenges:

- SOA infrastructure performance engineering
- federated identity support and overhead
- identity and authorization management (IAM) integration and overhead

Additionally there are heterogeneous service client limitations and incompatibilities, such as those related to WS-Security, SAML, client identity propagation, user credential availability and propagation, client-side WS-Policy enforcement, service SSO, trust infrastructure, and client-side service proxy.

We can label this set of infrastructure challenges as the "Service Interaction Problem Domain" (which is discussed in Part 2 of this article series).

The technical challenges in this problem domain are some of the most daunting in SOA solution engineering, yet ignoring these issues can introduce a host of operational and performance inefficiencies.

Misconception #4: "SOA Performance is Available Off-the-Shelf"

There is a common expectation that modern vendor SOA platforms have built-in support for high-performance service and service composition processing. This can naturally lead to the dangerous assumption that no further architectural

- Define, deploy and enforce sophisticated, context and content aware security policies that represent the complex structures and relationships of the business.
- Seamlessly extend a cohesive security strategy from the IT enterprise perimeter, through presentation layers, through the service security infrastructure and on to back-end systems.
- Efficiently disseminate new security policies and policy changes across multiple, disparate service producer and service consumer platforms.
- Process service security calls with minimal performance impact on service compositions.
- Easily assimilate user identities from external security domains.
- Abstract disparate service-client applications and platforms, as well as service-producer platforms, from the complexities of SOA security.
- Abstract much of the SOA security infrastructure (and all service producers and consumers) from the dynamics of rapidly evolving SOA industry security standards.
- Centrally manage all service security issues for both service consumer and service producer platforms.

work is needed to ensure or maintain a responsive and scalable service architecture.

However, the truth is that to actually enable true high-performance, it is crucial to take the following into account:

- As your SOA foundations grow, XML will become the life blood of your business and XML processing overhead becomes a growing, cumulative service performance problem.
- Cryptographic overhead will significantly increase with SOA growth, as will its associated performance impact.
- WS-Security, SAML, Kerberos and SSL all introduce significant processing overhead.
- Similarly, security policy evaluation (authentication/authorization) cycles also introduce a layer of performance requirements.
- The same goes for identity and authorization management (IAM), LDAP, WS-Trust, and Secure Token Service (STS) interactions.
- Feature-rich "service mediation" layers between service consumers and providers further add layers of runtime processing.
- Then there's the extra auditing and reporting processing that needs to be regularly carried out in support of SOA governance.

Every service call can introduce performance overhead issues, but with service-oriented solutions, it is the cumulative effect that we're most interested in because these solutions are comprised of compositions of services that are expected to perform as an efficient unit at runtime.

To fulfill this expectation, performance engineering must be a foundational part of service-oriented solution design. In its absence, it becomes too easy to get to a place where a mouse click here or a data transfer there results in unacceptable latency.

Misconception #5:

"SOA Security Architecture is Available Off-The-Shelf"

Another common yet flawed perception is that a complete SOA-compliant security architecture can be purchased as a product from a vendor. In reality, establishing a secure service-oriented architecture requires end-to-end design guided by an overarching architectural strategy that addresses a myriad of complex and interrelated issues, such as:

- identity and authorization management integration
- client identity and credential mapping/propagation
- abstraction of security complexity from service clients and producers
- client-side service proxy support
- client-to-service SAML, WS-Security, WS-* incompatibilities
- user authentication and SSO
- SAML/Kerberos token generation
- trust domain infrastructure
- service security context design
- service access authorization
- central policy management governing service and client-side security behavior
- service endpoint exposure
- governance infrastructure integration
- service host platform security integration

- back-end information system security integration
- security architecture performance
- operational scalability of security policies and architecture.

Most vendor products individually address a narrow set of these security concerns. A successful SOA implementation begins with understanding the security issues together with their relationships, inter-dependencies, impacts on infrastructure, as well as impacts on performance.

Success also requires an understanding of vendor community security offerings and their limitations (including potential conflicts that exist between them). Long before a commitment is made to buy service security products, a comprehensive strategy should exist that defines how security products and technologies need to be incorporated.

Misconception #6: "Programmers Will Define and Build Reusable Services"

Many strategic SOA investments are made with the assumption that developers will automatically begin decomposing workflows, business processes, database interactions, and presentation elements into reusable services. Although a nice daydream, it is rarely a reality. While developers certainly have the skills to build reusable services, their traditional motivations are usually geared to producing the opposite. The process required for molding automation logic into reusable services requires modeling, analysis, and design effort that results in a service being positioned as an IT asset and building block of a larger whole. All this translates into additional time and effort required before the services are actually developed.

Without external influence and a new development governance structure, developers will not (and essentially cannot) deliver reusable services on their own. This strikes at the core of SOA ROI expectations and the notorious "bad implementations". Without reusable services, services can be recomposed, and without repeated composition, the potential to attain true ROI and business agility fails to materialize.

Misconception #7: "SOA is a Process-Centric Architecture"

The SOA vendor community and industry analysts have generated volumes of literature concerning service orchestration, BPEL, and BPM. This can lead to perceptions that SOA is a process-centric architecture, which again is simply wrong.

While business process logic is always a part of a service-oriented solution, an SOA fundamentally relies on data. It requires an investment in information architecture to bring out its full potential.

Just as reporting and analysis software requires a higher level perspective on "business entities", so do service-oriented solutions and consuming transactional data services. There is a need to model higher-level transactional, operational business entities that processes and service-compositions can consume (something that vendors do currently provide tools for).

A modest investment in transactional, operational business entity modeling for SOA data services can yield a number of benefits:

- Process abstraction from data source instabilities, evolution and dynamics.
- Process abstraction from low-level data source interaction, integration and security issues.
- Process abstraction from multi-source data aggregation, collation, transformation complexities.
- Reusable, service-enabled transactional business information entities.
- Foundation for rapid data service evolution.
- Consistent, authoritative service-based data foundations for composite workflows, applications and process services

Misconception #8:**"SSL/TLS is a Good Foundation for SOA Message Security"**

The established use of SSL technology has resulted in the wide-scale use of SSL as a messaging security foundation for service-oriented solutions. SSL is an excellent means of establishing a long-term, secure session between a browser and a Web server; however, it is a poor foundation for securing one-time, session-less interactions between transient services.

It is the nature of workflow logic and service compositions to assemble business functionality by stringing together a set of programs (services, agents, etc.) along a message path, each of which may sequentially consume one or more messages.

Composite applications and service compositions are a key design goal of SOA engineering. By design, these types of systems and services "compose" high-level business functionality by orchestrating multiple calls to component services (and possibly to other composites). Since there is no session relationship (or even location relationship) between loosely coupled services, SSL makes these compositions highly inefficient. Each individual service call in the overall composite would need to negotiate and discard its own short-lived SSL session. Additionally, each service call may involve multiple segments of SSL along the message path between the service client and the service endpoint. Each of these segments represents a separately negotiated SSL session and adds to this cumulative inefficiency.

A WS-Security client/service interaction typically requires a single request and a single response message, both of which have business value. Additionally, most service calls might have one or two fields in a message that might actually be sensitive in nature. With WS-Security, only those two fields would be encrypted. With SSL, we must incur the cryptographic overhead of encrypting and decrypting every byte in both the request and response messages (after incurring the overhead of negotiating a short-lived session).

SSL overhead is cumulative with service compositions. It can choke the performance out of the best of service architectures while flooding your network bandwidth with volumes of session negotiation traffic that generates no business value. Although there may be times when SSL might be acceptable, good SOA design begins with avoiding the introduction of unnecessary inefficiencies in network bandwidth utilization, processing overhead and cryptographic overhead.

Misconception #9:**"All You Really Need are Web Services"**

We have occasionally encountered the perception that Web Services are all that is needed to generate SOA ROI. From this perspective, service-oriented technology architecture and infrastructure investments are secondary concerns and mostly the subject of vendor hype. A variation of this misconception is the mindset that "it's OK to build lots of Web services for now, because we can always invest in the required technology architecture later".

These are among the more seductive and damaging assumptions. They're tempting because it's easy and inexpensive to bang out Web services without strategic planning or investment. They're damaging for the following reasons:

- **Poor SOA Foundations** - With a focus on Web service development, service architecture is evolved rather than designed. The evolutionary drivers are typically narrow, near-term, vertical requirements rather than strategic IT goals. This inevitably results in scattered, fragmented, and poorly integrated service and security foundations that are ill suited to offer sustainable ROI.
- **Operational Inefficiency** - The proliferation of Web services across various platforms with no strategic architectural foundations inevitably leads to costly operational challenges in service security management, service deployment and administration, service scaling, client on-ramping and more.
- **Loss of Business Agility** - Take a minute to glance at the capability list that we began with in this article, and review the capabilities necessary to support and sustain business agility. These are complex, interrelated, inter-dependent requirements. Business agility won't magically evolve from a mess of Web services. It must be strategically planned and carefully engineered as a cohesive fusion of numerous complementary requirements and capabilities.

A decision to proliferate Web services without strategic planning or investment is a trade-off between modest near-term savings and high long-term costs. Inevitably, near-term gains give way to growing structural inefficiencies that will

prove costly to maintain and costly to remedy.

Conclusion: Evaluating SOA Engineering Directions

So far, our discussions have provided us with a fundamental understanding of the functional capabilities necessary within a successful SOA implementation. They have also provided an understanding of key engineering misconceptions that are often the root cause of SOA technical implementation problems. We close with a look at how IT leadership can apply this knowledge toward ensuring successful SOA adoption.

We begin by noting that the IT functional capabilities listed earlier need to be accommodated in some tangible manner within the architecture and infrastructure of a successful SOA implementation. This implies that each of these functional capabilities should map back to tangible architecture/infrastructure elements and strategies within the proposed SOA engineering solution.

Additionally, our understanding of common engineering misconceptions is also valuable. Armed with this knowledge, IT leadership can begin to evaluate proposed SOA engineering directions from a high level perspective.

The need to account for functional capabilities and avoid misconceptions within a service-oriented solution can be expressed through a series of questions that can be presented to SOA engineering teams. The following list summarizes these questions by category. Collectively, these questions can be used to gauge whether an SOA project is on track.

Questions relative to IT capabilities necessary to achieve increased ROI:

- How does each of the capabilities map back to SOA and/or enterprise systems or subsystems within the proposed solution architecture?
- Is there architectural depth in the proposed solution for delivering the SOA-related functionality behind each capability?
- What relationships, dependencies and/or interactions does each IT capability have with other aspects of SOA engineering. (And how are they accommodated within the service-oriented solution?)

Questions relative to common SOA engineering misconceptions:

- Is the proposed solution basically an SOA vendor product stack?
- Does there appear to be a predisposition to let a SOA product vendor define the solution architecture?
- Does the proposed solution architecture provide a comprehensive plan for integrating the IT enterprise assets listed in the *Misconception #1: "SOA Vendor Stacks are Service Architectures"* section?
- What is the selection criteria for the SOA architectural elements? (And, is there a clear selection justification?)
- Does an ESB product appear to be the essence of the proposed service infrastructure design?
- Is an SOA performance engineering strategy a key part of the proposed solution or is there simply an assumption that the vendor products will take care of that issue?
- Does the proposed security architecture seem to rely mostly on the capabilities of an SOA vendor security product?
- Does the proposed solution provide architectural support for integrating an SOA governance model that ensures the development of reusable services? (And how does the solution support governance?)
- Does the proposed solution effectively address the information architecture needs of the SOA initiative? (If yes, then how?)
- Is SSL a major part of message security within the solution architecture? (If so, why?)
- Does the proposed solution architecture offer any depth beyond service hosting platforms?

Ultimately, management needs to open the channels of communication with SOA engineering teams in order to challenge them to demonstrate their thinking relative to delivering key IT capabilities and avoiding misconceptions

within their proposed solution. Effective SOA architects then must also welcome the involvement of management and need to be willing to develop and evolve architectural solutions based on the strategies and issues covered in this article. The feedback provided can help management develop a working appraisal of the expertise and experience of the SOA engineers and then assess the likelihood of a successful SOA implementation.

References

[REF-1] "Top 5 Reasons SOAs Fail", Gartner, June 26, 2007

[REF-2] "Bad Technical Implementations and Lack of Governance Increase Risks of Failure in SOA Projects", Vektrel Solution Template Library and Knowledge Base (SOA Architectural Domain)

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008
SOA Systems Inc.