



THE TOP 5 SOA ADOPTION PITFALLS OF 2005

The Top 5 SOA Adoption Pitfalls of 2005

By: Thomas Erl, author of "Service-Oriented Architecture: Concepts, Technology, and Design"

It's been a tumultuous year in the world of SOA and we're just at the beginning of rollercoaster ride. As organizations continue to re-examine the ever-shifting landscape of service design, service buses, service governance, and even just services, there are often mixed emotions. Many are confused as to the maturity and overall status of SOA in the IT industry, but there is definite sense of excitement around its touted potential to unite business and technology.

Many SOA initiatives were launched this year, each with its own set of goals and expectations. Some failed miserably, while others failed just a little. For many, the determining factor in fulfilling their original objectives was drawing upon the experience of those who had already survived projects with less fortunate results. These individuals lived to tell their stories and warn others of what lies ahead along the path toward SOA.

In our line of work we get pulled into various projects at different stages of completion. We've seen good SOA go bad and bad SOA get even worse. Problems can be fixed and mistakes can be undone, but of course there is always an impact to getting things back on the right track. The best course of action is, obviously, avoiding problems and mistakes in the first place.

Understanding the pitfalls others have fallen victim to puts you in a position from which you can form the extent of foresight required to chart a safer route down your own SOA roadmap. To help you get a head start, we have collected the five most common SOA adoption pitfalls of 2005.

#5 NOT UNDERSTANDING SOA PERFORMANCE REQUIREMENTS

Loose coupling comes at a price. When implemented with Web services, SOA introduces layers of data processing as well as the associated performance overhead imposed by these layers. When starting out small, it is easy to build service-oriented solutions that function and respond as expected. As the scope increases and more functionality is added, the volume of message-based communication predictably grows. This is when unprepared environments can begin experiencing significant processing latency.

Critical to building a successful service-oriented solution is understanding the performance requirements of your solution and the performance limitations of your infrastructure ahead of time. This means testing (and, if necessary, strengthening) the message processing capabilities of your environments, and paying close attention to service design so as to achieve an acceptable balance between transmission rates, transmission sizes, and other service characteristics that can negatively affect solution performance.

#4 NOT STARTING WITH AN XML FOUNDATION ARCHITECTURE

In the world of today's SOA, everything begins with Web services. That statement has become a mantra of sorts within some organizations, but it is not entirely true. In the world of today's SOA, everything, in fact, begins with XML. It is the standard from which multiple supplementary standards have evolved to form a de facto data representation architecture. It is this core set of standards that has fuelled the creation of the many Web services specifications that are now driving SOA.

So much attention is given to how data is transported between services that the manner in which this same data is structured and validated behind service lines is often neglected. This oversight can lead to an improper implementation of a persistent XML data representation layer. This layer is foundational to SOA and its weaknesses will have repercussions across any solutions built upon it.

#3 NOT CREATING A TRANSITION PLAN

The chances of a successful migration will be severely diminished without the use of a comprehensive transition plan. Because the extent to which service endpoints are positioned within an enterprise can lead to a redefinition of an environment's infrastructure, the affects of a poorly executed migration can be significant.

Transition plans allow you to coordinate a controlled phasing in of service-orientation and SOA characteristics so that the migration can be planned on a technological, architectural, and organizational level.



Typical components of an SOA transition plan include an impact analysis (that predicts the extent of change SOA will impose on existing resources, processes, custom standards, and technology), transition architectures (that map out a series of planned hybrid stages toward a target SOA), and a speculative analysis (that takes into account the future growth of Web services and supporting technologies).

#2 NOT STANDARDIZING SOA

SOA, like any other architecture, requires the creation and enforcement of internal design standards for its benefits to be truly realized. For example, if one project builds a service-oriented solution in isolation from others, key aspects of its solution will not be in alignment with the neighboring applications it may be required to interoperate or share agnostic services with one day.

This can lead to many problems, including incompatible data representation, service contracts with irregular interface characteristics and semantics, and the use of non-complementary Web services extensions (or extensions being implemented in different ways).

SOA promotes a development environment that abstracts back-end processing so that it can execute and evolve independently within each application. However, standardization is still required to ensure consistency in design and interaction of services that encapsulate this back-end logic.

#1 BUILDING SOA LIKE TRADITIONAL DISTRIBUTED ARCHITECTURE

The number one obstacle organizations have been facing when attempting to achieve SOA is building service-oriented solutions in the same manner in which traditional distributed solutions have been built, under the pretence that SOA is actually being achieved.

SOA is not CORBA + XML, nor is it ASP.NET + WSE. Service-orientation is not object-orientation, nor is it "close enough" so that building object-oriented component logic will always "fit right into" service-oriented solution environments. SOA is a distinct architectural model based on service-orientation, a distinct design paradigm. Understanding these distinctions is critical to building automation logic that is truly service-oriented and in alignment with where the SOA industry is moving on a global scale.

This article contains excerpts from "Service-Oriented Architecture: Concepts, Technology, and Design" by Thomas Erl (792 pages, Hardcover, ISBN: 0131858580, Prentice Hall/Pearson PTR, Copyright 2005). For more information, visit www.soabooks.com.

Thomas Erl is the world's top-selling SOA author. His first book, "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services", provides strategic guidance to establishing service-oriented integration architectures. His second title, "Service-Oriented Architecture: Concepts, Technology, and Design", is the first "how-to" guide to building SOA, and documents step-by-step processes for service-oriented analysis and service-oriented design, as well as comprehensive coverage of service-orientation principles. For more information, visit www.soabooks.com.

Thomas is also the founder of SOA Systems Inc., a research and consulting firm specializing in SOA consulting, planning, and training services. SOA Systems has made significant contributions to the SOA industry in the areas of service-orientation research and the development of a mainstream SOA methodology. For more information, visit www.soasystems.com.

Thomas is a voting member of OASIS and also participates as a speaker and instructor for private and public events and conferences.

Contact: terl@soasystems.com

See the original article at:

<http://www.bpminstitute.org/articles/article/article/the-top-5-soa-adoption-pitfalls-of-2005.html>