

The SOA Magazine

Feature Article



Automated Modeling and Performance Management for Service-Oriented Solutions

by Chris Farrell

Published: April 3, 2007 (SOA Magazine Issue VI: April 2007, Copyright © 2007)

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

Abstract: Gartner recently reported that 40 percent of service downtime results from application-related failures. Others are predicting that performance issues will be more than 50% higher in service-oriented applications due to increased levels of design complexity. As a result, the ability to effectively manage and monitor application performance is emerging as a new critical business imperative. With the mass adoption of SOA and the wide-spread deployment of enterprise service-oriented solutions, what steps must companies take to reduce application downtime?

Traditional application performance management (APM) methods don't adequately address the constantly-changing nature of distributed environments. They simply don't take into account the layers of complexities that underlie multi-platform, multi-vendor, and even multi-company solutions. This article will examine the concept of "modeling" an application's architecture in support of its eventual performance management. It will also explore how automating the APM process for service-oriented applications can reduce the risk and effort associated with the management of application performance.

Introduction: The Value of SOA

With so many organizations implementing various types of service-oriented applications, SOA has matured into more than just an architectural theory. It has established a philosophy and an attitude that is revolutionizing the way business and IT teams work together. So regardless of your organization's current plans, one thing is for sure: if you are in IT, SOA will be part of your life in the future.

With technology offerings from companies like BEA, IBM and Microsoft, customers are implementing SOA-based applications in production, all hoping to attain promised strategic benefits, such as increased application flexibility, agility and reuse. In fact, enterprises of all sizes are increasingly applying service-orientation and supporting technologies and are thereby effectively establishing a new generation of mission critical business applications. The kinds of solutions are varied, ranging from "simple" portals to complex, feature-rich, and highly integrated applications. The typical driving force behind SOA initiatives is the desire to improve IT's ability to deliver on-demand, more salient business automation for the right employees, partners, and customers.

Well planned and executed SOA projects have the potential to turn around many of the issues concerning IT organizations today, including lack of interoperability, schedule delays, budget over-runs, and the overall perception that IT cannot effectively respond to business needs. SOA also provides tactical paths to creating economies of scale for mergers, acquisitions, spin-outs, spin-ins and other unique organizational relationships. If you are not currently carrying out an SOA strategy, chances are that you have started to look at one.

Service-Orientation and Black Box Engineering

SOA's main premise revolves around how application functions are designed, built and implemented as they relate to the business processes they're supposed to serve. Each business application is broken down into discrete services. Rather than create a single application that contains all the necessary services, each particular service is built and

deployed on its own, allowing massive reuse and flexibility as business needs expand and change. This allows the business applications themselves to remain relatively stable, while individual services can (and usually do) change as often as necessary. Change can be required to deal with business problems (such as acquiring a new division) and technology problems (such as adopting a new identity management model).

Another benefit to building service-oriented solutions is that some of the required services can be “bought” from third parties, either for internal deployment or on-demand runtime requests. This helps avoid having to create each and every piece of a new application.

Thus, SOA extends beyond vendor neutrality and builds on platform neutrality. The service-orientation design principle of Service Abstraction is analogous to the black box programming concept. The platform, technology, and programming inside each service are hidden from its potential consumers. While black box engineering provides increased flexibility when designing large projects across multiple organizations, this approach always comes at a price – the inability to easily monitor and manage application performance. Some experts have compared the experience of managing black boxes to seeing inside a black hole.

Like middleware projects a decade ago, application performance management (APM) is a primary factor that can slow down SOA deployments. The rest of this article explains how IT organizations can move ahead with both the deployment and subsequent management of service-oriented solutions while accommodating the black box factor.



Figure 1: As service-oriented applications continue to become more complex, performance management becomes an increasingly important issue.

A Look Back at Application Management History

The largest problem with any monitoring project, from network to application performance, is determining what to monitor. In the field of application management, the issue of knowing what to monitor gets harder because there are multiple levels of potential measurements. The business processes and transactions are the reason for an application’s existence, but knowing them alone isn’t enough. To achieve effective application management, it’s critical to correlate the internal application components to the overlying processes and transactions. To understand the logistic issues associated with this, a look at APM history is relevant.

As enterprises first began using Web technologies to make data and transactions available to end users, the applications were fairly simple. For example, a Web solution may have been required to access a piece of data and then present this data to the user via a browser. A more complex business process was often divided up into multiple individual Web pages, each running a separate back-end component or application that performed a discrete function. Overall, these applications were easy to monitor.

As the benefits of Web solutions permeated businesses, the application server became a common part of application architecture. All of the business logic could now be put into a single application instead of creating one for each page request or process step. The application essentially acted as an integration hub, bringing together the required back-end data and transaction requests into one place and delivering the consolidated results to the requester.

This is where things got tricky. Java and Java Virtual Machines (JVMs) made it difficult to see what was happening at the point of integration. A set of solutions sprung up reporting on proprietary data inside the JVM and providing detailed information about the point of contact. While the ability to see that data was critical, the data itself proved to be both a commodity and a burden.

Additionally, having the data didn’t solve the dilemma of deciding what to monitor. This became a larger issue as

enterprises started dealing with more complex applications, quicker turns of application versions, and the introduction of new technologies (such as portals built around a service-oriented architecture).

Implementations typically required weeks of reverse engineering and involved professional services consultants, senior developers, and architects. Worse, the most common process was simple trial and error, with the hope of determining what deep application metrics correlated to the overlying processes. To make things more complicated, the operations team also had to find the proper mix of appropriate problem resolution and overall application overhead. When another application needed to be managed (or if a change occurred to the existing managed application) the entire exercise had to be repeated. While not efficient, the process could at least work for stable, hub-like integration architectures.

Service-Oriented Application Management Challenges

Today, applications have become more distributed – first with the introduction of process integration platforms, and then with increasing roll-outs of modern-day, service-based implementations. The manual correlation used for hub-based solutions is not practical for distributed applications. In fact, service-oriented application management in this way is literally impossible. With too many integration points and transaction types and relationships, application performance management decisions would have to revolve around millions of potential entities.

Let's look at an example in the service industry. A clearing house company runs an application for connecting auto insurance companies with service providers (and vice versa). The distributed application is literally comprised of two standalone applications that work in opposite directions, each running on a different J2EE platform. In between these two solutions reside more than 60 discrete services (with each service running as a separate, autonomous black box programs). While creating operational efficiency for IT logistics, the application has so many individual moving parts, that application management becomes a daunting proposition.

Managing this type of distributed environment requires an application management solution that understands the unique characteristics of distributed applications, especially in relation to the rate of change. Because manual correlation is no longer possible, the only way to effectively deal with this problem is through automation. And, the most effective way to insert automation into technology is through modeling.

An Introduction to Automation through Modeling

Automation helps solve several problems associated with SOA performance management, especially the aforementioned issue of figuring out what to actually monitor. An easy way to explain the process of APM is to discuss three specific stages:

1. Setup / Installation

- installation and configuration of monitoring agents
- creation of meaningful UI visualization

2. Analysis / Reporting

- correlation of business processes to internal application components
- root cause analysis

3. Change Management

- configuration of monitoring agents
- correlation of application details to business processes

Automated modeling during setup and configuration allows project teams to deploy managed applications faster and easier. Management tools with modeling provide a programmatic representation of the application architecture, allowing for the auto-correlation of deep application data with the overlying business processes. This also helps with the selection of deep monitoring locations and the strategic insertion of corresponding agents.

Having the complete correlation between business processes and deep data allows modeling-based tools to automatically create user interfaces around service level reporting, performance review, and root cause analysis.

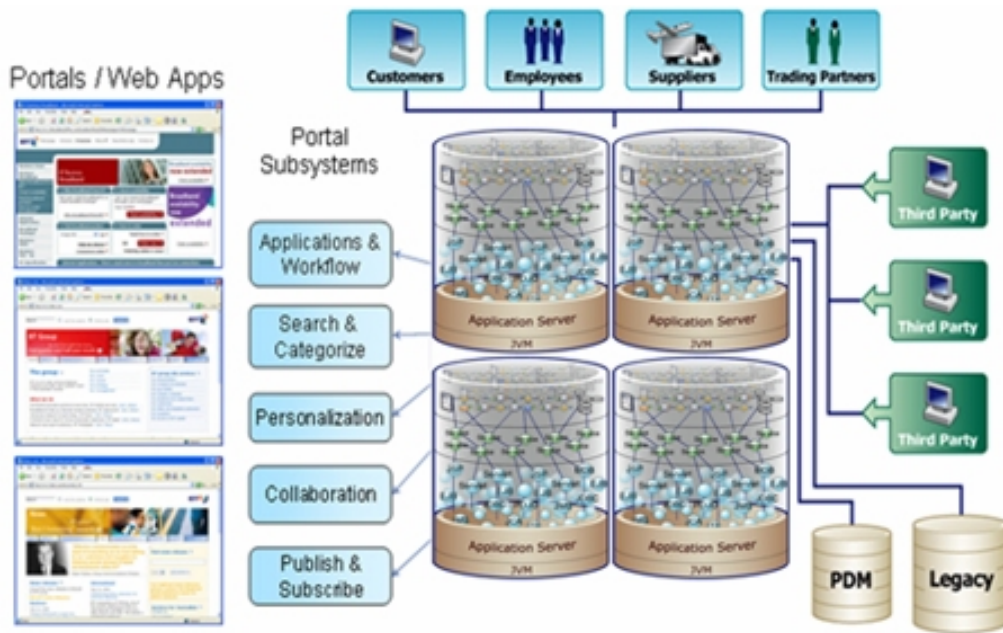


Figure 2: Modeling allows an APM solution to organize the deep application component metrics in an appropriate manner within the context of how services relate to overlying business processes – and each other.

APM analysis processes (especially those associated with root cause analysis) have traditionally been a manual exercise, carried out with traditional data-centric tools focused on application servers. Problem resolution activities for these tools typically revolved around leveraging an expert (senior developer or architect) to interpret the mass of the retrieved data, hoping that it will actually be useful.

Modeling helps automate problem triage by starting with an architectural representation or an application schematic. This allows for a quick diagnosis down the architectural path of a business transaction to isolate root causes. An additional benefit of modeling is that once a problematic component is found, application owners can “flip the context” and see all impacted business processes based on discovering the problematic service.

While change management is always something of concern, in SOA environments it is critical. For example, a relatively conservative estimate of change frequency for a service is, say, one change per year. Doing the math, that translates into more than one change per week on average. If quarterly changes are allowed, that essentially results in a change per day. Manually adjusting to changes of this magnitude is not only inefficient, it is simply not possible in a scaled business environment.

Modeling allows the APM solution to handle change management automatically. This process begins with the automatic detection of changes in the application environment and then proceeds with the automatic adjustment of the correlation between processes and application components. You can subsequently decide what to monitor and then deliver a new application schematic as required.

Looking Ahead

Service-oriented architecture lies at the heart of the powerful service-oriented computing platform. It allows IT departments and their related business organizations to work more efficiently and responsively. As service-oriented applications become more and more prevalent, the ability to effectively manage them will become increasingly critical. There are specific steps organizations can take to make SOA management work for them:

1. *Make Management a Priority* - Management shouldn't be an afterthought. Make sure that it is budgeted for when pulling together SOA projects – both in time and money.
2. *Make a New Decision* - Managing SOA is a new problem, one that deserves true evaluation. You might decide to use one of your existing APM vendors, but don't just write them in. SOA presents unique challenges that may be difficult to accommodate with traditional tools.
3. *Start Small* - Service-orientation introduces a change in design philosophy which, in turn, introduces some cultural challenges. Don't add to this impact by trying to run multiple applications out of the gate. Like other technology breakthroughs, the chances of success increase when dealing with a single problem at a time.

4. *Assume Nothing* - Don't assume that what you deploy is architected and coded the way you originally designed it. Get runtime information to help determine what the actual production architecture looks like.
5. *Communication, Communication, Communication!* - Involve all of the key stakeholders, and often. It is important for executives to see the value in the project, for line-of-business managers to see that their applications will run better, for IT management to witness efficient operations, and for developers to realize their innovation potential. Keep them all in the loop from start to finish.

Conclusion

The governance of any new application platform will encounter barriers, and the management of service-oriented solutions is no different. However, once the mystique has faded and you begin to understand how to monitor services and what specific aspects of service performance to monitor, then rolling out managed applications as needed becomes much more... manageable.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL

