

## Service-Oriented Architecture Scenario

Yefim V. Natis

Attempted SOA will cause great successes and great failures of software projects. Understanding its role and meaning, beyond the simplistic hype, is the imperative for every enterprise software architect.

## ANALYSIS

---

Service-oriented architecture (SOA) was first described by Gartner in 1996 (see *SSA Research Note SPA-401-068*, 12 April 1996, "Service Oriented Architectures, Part 1" and *SSA Research Note SPA-401-069*, 12 April 1996, "Service Oriented Architectures, Part 2"), but the recent interest in the architecture has been spurred by the emergence of a powerful industry trend: Web services. Although Web services do not necessarily translate to SOA, and not all SOA is based on Web services, the relationship between the two technology directions is important and they are mutually influential: Web services momentum will bring SOA to mainstream users, and the best-practice architecture of SOA will help make Web services initiatives successful.

### **SOA and Web Services: Two Complementary Talents**

The definition of "SOA" is discussed in detail in "Introduction to Service-Oriented Architecture." Essentially, SOA is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls. SOA would be better-named "interface-oriented architecture." SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function. Services are software modules that are accessed by name via an interface, typically in a request-reply mode. Service consumers are software that embeds a service interface proxy (the client representation of the interface). Web services are defined elsewhere in Gartner research (see "The Web Services Provider Platform: You Already Have One" and "Introducing Common Sense to Web Services"). Simply speaking, any software that uses the standards Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) or Universal Description, Discovery and Integration (UDDI) is a Web service.

As evidenced by the definitions, Web services are about technology specifications, whereas SOA is a software design principle. Notably, Web services' WSDL is an SOA-suitable interface definition standard: this is where Web services and SOA fundamentally connect. Those who see Web services as architecture regard WSDL as the definitive standard of Web services (others see SOAP as a definitive standard for Web services — this is a view of Web services as a communication method). In practical use, the ubiquitous Web services standards enhance the mainstream appeal of SOA design. *Through 2008, SOA and Web services will be implemented together in more than 75 percent of new SOA or Web services projects (0.7 probability).*

### **The Best Use of SOA Is for New, Multichannel and Composite Real-Time Applications**

SOA is gradually replacing monolithic architecture as the premier design principle for new business applications. This process is driven in part by the inherent benefits of SOA for new application projects (see "Predicts 2003: SOA to Stir Up Application Server Market"). However, several recent developments in the software industry have helped to bring SOA further to the forefront. With personalized client/server, Web-based and portal-style user interfaces, an increasing number of projects require reuse of application business logic over multiple access channels. Different user categories (operators, customers, mobile sales staff, self-service employees, managers), in different situations (office, home, road, hotel) and using different devices (personal digital assistants, phones, laptop computers) all may request access to the same essential set of back-end business functions. The loosely coupled SOA provides the natural basis for unintrusive reuse of the back-end logic by multiple styles of clients. Thus, the transition to multiclient and multichannel applications naturally pushes forward the SOA-based application design (see "Predicts 2003: SOA Comes of Age via Web Services").

Another fundamental trend in application engineering is the reuse of older and external applications in new user-facing transactions. The composite application style has emerged as an

alternative to the development of all-new applications. In fact, in 2003, the majority of new business applications have some connections with external resources and thus are composite applications. Many projects have created programmatic interfaces to wrap legacy and other external functionality for assembly into heterogeneous composite transactions. Composition and integration of old and new business components into new real-time transaction patterns are a natural fit with SOA.

Because composite applications deliver a form of integration, a myth has emerged in the industry that SOA and Web services are the modern answer for the application integration problem. In reality, SOA is indeed helpful in some cases of integration (real-time composite transactions), but it is less useful in others (see "Use SOA for Composite Application Integration"). Even in the case of the composite SOA transactions, additional integration technology is typically required to reconcile the information model differences of the participating applications (see "Most Composite Applications Will Need an Integration Layer").

### **Ironically, Users Who Best Understand Event-Driven Architecture Will Be the Best Users of SOA**

A large part of the integration problem requires implementation of long-running offline business processes. Here, event-driven architecture (EDA), not SOA, is the industry best practice. Unlike SOA, EDA is the design vision for long-running asynchronous processes (SOA is best applied to real-time request/reply exchanges). In EDA, a process node posts an event (in SOA, a process node makes a targeted processing request). In EDA, posting an event reflects results of some *past* processing (in SOA, making a processing request directs *future* processing). In EDA, the poster of the event is disconnected from the processors of the event, if any (in SOA, the requestor of service knows the service and depends on its existence and availability).

The nature of data retrieval transactions fits well with the model of SOA (make request for information, wait for the reply, disconnect on receipt of the reply). The nature of update transactions fits well with EDA (request the update, ensure delivery of the request, release the resources without waiting for the time-consuming process of applying the updates). The nature of composite transactions fits well with SOA (represent a complex real-time process as a single transaction, hide the complexity of composition and integration behind the wrapper interface). The nature of multistep processing fits well with EDA (monitor status, trigger processing based on changes of status, evolve a process through its component steps as status changes from initiation to completion).

Look-ups and updates of simple and complex data, new and composite real-time transactions, and multistep asynchronous processes, mentioned above, all represent aspects of modern application design. Most projects are best-served using in part both architectures and their respective enabling technologies. Thus, the key to best use of SOA is in understanding its context and its alternative — the EDA. Deploying the less fitting architecture may still allow successful project completion, but it is likely to underpower the application and to leave the enterprise with the competitive disadvantage of less-agile IT.

### **Enterprise SOA Is Pragmatic SOA: The Best Price-Performing Protocols Win**

Web services technology is widely accepted, and some of it is ubiquitous. Yet, its power is limited. First, Web services, like a Trojan horse, will bring SOA into often-unaware enterprises in the form of simple, often opportunistic, service-oriented applications. Second, once enterprises begin to realize the benefits of SOA, the enterprise computing reality will force users and vendors to new technology practices. Web services standards will be expanded to support greater levels of quality of service. SOA middleware and SOA tools will be expanded to optimize the middleware operations and to allow substitution of protocols and technologies, with the goal of preserving the core benefits of SOA while eliminating the overhead of ubiquitous, but simplistic,

standards. *Through 2008, multiple protocols and multiple technologies will be involved in enterprise implementations of SOA despite the wide adoption and gradually increasing maturity of Web services standards (0.8 probability)* — see "Picking the Right Interoperability Strategy for SOA."

### **For Most Enterprises, SOA Will Arrive via SODA Tools and via Updated Application Packages**

Design and development of SOA applications are inherently systematic (see "Systematic vs. Opportunistic: Useful Heterogeneity"). It requires investment of advanced architectural thinking into definition of services before any development of services or service consumers can begin. Earlier SOA-style technologies — notably Common Object Request Broker Architecture (CORBA) and XATMI — have failed to be adopted by mainstream projects because, in part, of the demanding design process that SOA requires. This issue is not eliminated by any of the modern SOA or Web services middleware. SOA remains difficult and risk-prone (see "Users: Beware of Opportunistic Web Services Projects"). The difference-making news now is the emergence of SOA-dedicated development tools. These, in most cases, are focused on the Web services "flavor" of SOA, but they will expand over time to cover alternative protocols as well. Most users will begin their adoption of SOA through adoption of modern development tools. When developing an SOA application becomes easier than developing a monolithic application, the decisive majority of the software industry will switch to SOA. The key to this transformation is the availability of great tools (see "Next-Generation AD Tools Will Bring SOA to the Mainstream").

Most enterprises prefer to buy software solutions when they can find them, rather than build them in-house. Once bought, the application packages must be maintained and integrated with the rest of the enterprise portfolio of applications. To facilitate this process, most application independent software vendors are wrapping or converting their applications to SOA. New vendors are beginning to emerge that design their application packages as libraries of SOA-style services from the ground up. *By 2008, more than 75 percent of then-current application packages either will be natively SOA or will expose SOA interfaces through a wrapping layer of interfaces (0.8 probability)*. Many enterprises that prefer to stay away from the leading-edge development styles will get their first exposure to SOA through buying the packaged applications and applying the composition capabilities of their portals or other composition-enabled products (see "Packaged Applications Meet Service-Oriented Architectures").

### **Agile Business Process Management Will Benefit From SOA, and Then, Underpowered, Will Pull EDA as Well**

Standards-based service interfaces will facilitate automation of business process management and design. New tools, capitalizing on the growing number of available service implementations, will support rapid business process orchestration with minimal new development (see "SOA and BPM Form a Potent Combination"). Composite application development will be the primary beneficiary of this process, as the SOA interfaces will operate best in a real-time request-reply mode. Increasingly, demanding business process engineering projects will inevitably reveal limitations of SOA in process management and will lead to increasing adoption of EDA and of the people-based workflow — alongside the real-time SOA-style process composition. Business process management technology will emerge as the crucial link in extending the momentum of SOA to EDA and, thus, in completing the fundamental architectural pattern of modern, agile enterprise IT: *It is service-oriented and event-driven*.

### **The Main Benefit of SOA Is the Opportunity for Incremental Development, Deployment, Maintenance and Extension of Business Applications**

Many myths have developed in the industry around SOA. The reality is more modest, but also more immediately beneficial than the hype. SOA brings these benefits to enterprise IT:

- Incremental development and deployment of business software
- Reuse of business components in multiple business experiences
- Low-cost assembly of some new business processes
- Clarity of application topology

SOA does not bring these mistakenly attributed benefits:

- Simple software engineering
- Free integration or interoperability
- Technology independence
- Vendor independence
- The ultimate architecture for the modern enterprise

Through 2007, growing enterprise experience with SOA process and SOA-based applications will eliminate the myths and instill appreciation for the real benefits of SOA in most enterprises. Evolving tools, skills and best practices will make development of SOA-style applications easier than development of monolithic applications. This change will shift the massive software industry mainstream into the new software-engineering reality: *By 2008, SOA will be a prevailing software-engineering practice, ending the 40-year domination of monolithic software architecture (0.7 probability)*. "Prevailing," however, does not translate to "exclusive." *Through 2008, most enterprises will combine elements of SOA, EDA and monolithic architecture in their enterprise software development projects (0.8 probability)*.

SOA is a good practice for software design. Although it is not an answer to all problems, SOA is useful and should be part of most modern software projects. Over time, lack of SOA will become a competitive disadvantage for most enterprises. Mainstream enterprises should invest today in understanding SOA and building SOA design and development skills.

## Features

"Introduction to Service-Oriented Architecture" — An understanding of the strengths and limitations of service-oriented architecture will help users to identify its role in the overall architecture of modern enterprise IT. **By Yefim Natis and Roy Schulte**

"Use SOA for Composite Application Integration" — Service-oriented architecture will be the best solution for composite application integration, but not for common unidirectional integration scenarios that involve data consistency or multistep processes. **By Roy Schulte**

"Most Composite Applications Will Need an Integration Layer" — In the near future, despite service-oriented design practices and Web services standards, most composite service-oriented architecture applications will use an integration layer. **By Roy Schulte and Yefim Natis**

"Picking the Right Interoperability Strategy for SOA" — No single interoperability platform for linking service consumer applications with services will emerge in the near future, despite the wide popularity of some Web services standards and protocols. **By Massimo Pezzini**

"Next-Generation AD Tools Will Bring SOA to the Mainstream" — New toolsets based on service-oriented development of application technologies will help deliver applications based on service-oriented architectures. **By Mark Driver**

"Packaged Applications Meet Service-Oriented Architectures" — Enterprises evaluating packaged applications will benefit from closely examining how well those applications can participate in a service-oriented architecture. **By Brian Wood and Jeff Comport**

"SOA and BPM Form a Potent Combination" — Business process management and service-oriented architecture will combine to benefit IT professionals and business users. **By Jim Sinur, David McCoy and Jess Thompson**

### Recommended Reading and Related Research

"Users: Beware of Opportunistic Web Services Projects" — Service-oriented architecture built opportunistically and at as low a cost as possible will be a disaster for enterprises' software infrastructures. **By Yefim Natis**

This research is part of a set of related research pieces. See "Service-Oriented Architecture: Mainstream Straight Ahead" for an overview.

## REGIONAL HEADQUARTERS

---

Corporate Headquarters  
56 Top Gallant Road  
Stamford, CT 06902-7700  
U.S.A.  
+1 203 964 0096

European Headquarters  
Tamesis  
The Glanty  
Egham  
Surrey, TW20 9AW  
UNITED KINGDOM  
+44 1784 431611

Asia/Pacific Headquarters  
Level 7, 40 Miller Street  
North Sydney  
New South Wales 2060  
AUSTRALIA  
+61 2 9459 4600

Latin America Headquarters  
Av. das Nações Unidas 12.551  
9 andar—WTC  
04578-903 São Paulo SP  
BRAZIL  
+55 11 3443 1509