

The Well-Managed Web Service

Herb VanHook, Vice President of Corporate Strategy, BMC Software

This paper presents imperatives and suggestions for successful operational management of the emergent technologies represented by the adoption of Web Services. Whether used to build SOA-based (Service Oriented Architecture) applications, or for basic application integration, the focus is on the run-time management of production applications.

The next few years will be marked by significant trends in the packaged and custom application market. These include:

- Alignment, convergence and consolidation among application logic, application integration options, and application servers.
- Adoption of service oriented architectures (SOAs) to design, architect and implement business applications as a set of *discrete business services*.
- Adoption of Web Services as the preferred standards-based integration technology for SOA-based (and non-SOA) applications.
- Increased use of programmable business process languages to describe business processes, workflows and services – and as a macro driver for SOA implementations.

These developments will enable a more direct modeling of the business environment and its processes, with the creation of technology objects that model the discrete business services that make up a company's business processes. SOA design principles, enabled through Web Services, will drive this new wave of enterprise software architecture, with business processes realized through loosely coupled composite applications.

Complete management of this emerging environment will depend on capabilities (discovery, security, availability, performance, change, etc.) at *all* layers within the technology stack and at all points in the operational process lifecycle.

Service-Oriented Architectures (SOA)

SOA is actually an old concept that has been “made new” by some emerging technology options. The term “SOA” expresses a software architectural concept that defines the use of *services* to support the

requirements of software users. SOA principles provide a methodology and framework for documenting business capabilities (processes and workflows) and translating them into models for technology implementation. SOA is not technology – it is technology independent.

In an SOA world, *service providers* (a node or software on a network) make resources available to *service requestors* (another node or software on the network) as independent *services* (e.g., perform a function, obtain some data). A *service* is a self-contained, stateless business function (e.g., verifying a credit card) that accepts one or more requests and returns one or more responses through a well-defined, standard interface.

While most definitions of SOA identify the use of *Web services* (see below) in their implementations, SOA implementations may actually use a variety of technologies. With SOA, software applications integrate with one another in a *loosely coupled* manner. This means the *service requestor* and *service provider* can be implemented using completely different technologies on different infrastructure platforms, as long as they obey the rules of the standard interface.

Web Services

Web Services are a set of interface standards (message formats, protocols, etc.) to provide interoperability between software applications running on disparate platforms. Web services are used to integrate software running on different machines. The protocols and data formats are text-based where possible.

Web Services, along with SOA, are driving a new form of application architectures that is being used by both packaged application vendors as well as custom

application developers. Web Services are the current *standards-preferred* method for implementing SOAs. However, Web Services are also used for basic application-to-application integration needs (independently of any SOA architecture). Some of the standards in Web Services derived from protocols and formats that originated within the World Wide Web; therefore, the term “Web” stuck in the name.

The Rise of the Business Service

IT and Business alignment (common goals, common vocabulary, common metrics, etc.) is desired to maximize the value of information technology, and to understand and manage the opportunities, risks, costs and impacts of a technology-enabled business. By using Service Oriented Architectures (SOA) to implement technology models of business services and processes, a new focal point (the discrete business service) is emerging to enable this alignment, with business processes being composed from multiple business services.

In the broader sense, a “Business Service” can be anything the business defines it to be. However, in the context of SOA, it takes on a stricter definition. In this case, a *Business Service* is an *interoperable software component that provides a business function, and models that business function as it exists in the “real world” business domain.* Examples of business services are “add a new customer,” “perform a credit check” and “approve an order.” The software components represent a digital model of the business. SOA design principles and integration standards (e.g., Web Services) enable creation of these software components. However, even with adoption of SOA-based applications, businesses will still run (and depend on) a mixture of other application architectures (monolithic, client/server, etc.).

New applications will be developed with SOA principles, and legacy applications will be updated to present SOA interfaces – thereby modeling real-world business services (i.e., a particular business task). Packaged applications will be a set of related Business Services and the application infrastructure that will run them. Application code bases for business service development will be varied, but new development will primarily be J2EE & .NET.

Software that is designed and developed with SOA models causes the discrete business services to become “real” and discrete technology components (i.e., they now exist as a clearly identifiable physical presence within the technology stack).

Even without explicit SOA-class Business Services, the business service can be modeled (represented

and virtualized) by abstract mapping to a discrete application function, a discrete transaction, a complete business process, a collection of supporting infrastructure elements, and many other methods. However, a unique characteristic provided by SOA-based software applications is that well-defined models of the business services are now *automatically* available to operational management tools and processes.

Therefore, the *business service* is a technology abstraction of something that exists in the business domain (a part of one or more business processes), and it is also a service abstraction “container” for lower-level infrastructure and application elements.

Web Services Adoption

Web Services protocols and formats are emerging as the standards-preferred way to implement federated integration in distributed applications, and more specifically in service-oriented application software architectures. In this context, federated refers to the rules and policies that apply to each Web service as a stand-alone entity (its “self government”), but an integration of multiple services can also be governed by a higher level workflow, orchestration and choreography function (e.g., a business process engine becomes the higher-level authority and part of the “joining” infrastructure). The use of Web Services as an integration method (even without an SOA structure) is becoming mainstream, and maturity models and roadmaps are emerging that enable best practice adoption strategies.

A Web Service is not only an integration point within an application, but the Web Service itself becomes a representation of the business logic delivered by the executable code behind the interface. Whether an organization is using base-level Web Services merely as a new application integration technique, using them as a way to expose key business services externally to partners and customers, or using them to implement a top-down SOA design – the resulting implementations must be managed in the production environment. One of the biggest results of Web Service adoption will be the shattering of application silos, as reusability and repurposing of business logic occurs across internal and external boundaries.

As with all new technologies, this rapid adoption will incur its own management and operational challenges. As applications built using the enabling protocols, formats and identifiers come on board, the historical challenges of availability, performance, throughput, process governance, security, orchestration, optimization and control will prevail – albeit focused on a new technology model.

Additionally, Web Services also holds the promise of a new integrated management model, whereby the applications (and their enabling infrastructure) natively expose information and control points that can be leveraged by management tools through a Web Services interface. A number of emerging Web Services-based management standards address this possibility, and IT organizations will increasingly leverage the flexibility this approach promises.

However, current adoption of SOA principles and Web Service implementations are still early in the market. While a few organizations have made serious long-term investment, most companies are only using Web Services for some basic application integration needs or for a singular project.

Visualizing the Business

IT organizations have long wanted to view their infrastructure topologies – from physical layouts of infrastructure resources, through logical topologies of communication networks, all the way up to the representations of business process models and workflows. This will be no different in the SOA and composite application world. Users will want to be able to graphically display their Web Services environment, showing the services themselves, their logical connections (service providers and service consumers) and the overall composite map.

Indeed, in the true SOA-based enterprise, these visualizations provide a true proxy of business process models, and are therefore a critical alignment point between technology organizations and the business groups they serve. Whether provided by an automated discovery exercise or other means, these topology relationship maps will be common tools in the management of Web Services.

The capability of visualizing a business model (or business process) as a “service net” is compelling. This will carry the most weight and impact in organizations where top-down SOA design has happened.

The Basics: Availability, Performance, Throughput

Although Web Services represent an interconnection point within an integrated flow, they also are a distinct entity in providing the service embodied in the business and application logic “behind” this interconnection point. Some of the fundamental things desired in managing a Web Services environment are answering the following basic questions about each Web Service resource:

- Is it available for use (i.e., is it accessible and responding to requests)?
- Is it working well (i.e., is it functioning as intended)?
- How fast is it working (i.e., how much work is going through the interface, what are the throughput measurements)?

For most new resources introduced into an IT infrastructure, these same questions apply. Users will use both passive and active methods for capturing this information – from synthetically invoking the Web Service to see how it responds, to local monitoring of the Web Service to observe its real-time behavior. This information provides early warning of availability and performance problems, as well as providing input on rapid service adoption, service usage and load, and input for related processes such as capacity planning and service level reporting.

One of the end goals of Web Services adoption is to achieve a high rate of reuse of common functions. If this succeeds, users will have multiple-application dependencies on the centralized functionality represented by a single Web Service. If a commonly shared Web Service is unavailable or is performing poorly, it has the capacity to cascade problems across an enterprise or to external service users.

Many users today have tools to measure and track transactions. These must also work well with and through composite application environments, to ensure problem isolation and root-cause efforts can be tackled. Even though the interconnections in a Web Services world may be loosely coupled, the transaction experience of an end user still appears seamless.

Application Problem Resolution

For application problems involving Web Services, many of the diagnostic and root cause tools in use today do not give sufficient insight into the right level of the technology to discern the problem. These tools must have knowledge of Web Service protocols and formats, understand the typical errors that may result (e.g., de-serialization errors), and lead application experts along the right path for resolution.

Any process and/or workflow tools in use today must keep pace with the evolution of application problems that will be presented by use of Web Service technology. The modularity of these architectures coupled with the high rate of change is going to cause organizations to revisit (and perhaps redesign) their problem management processes. Problem management systems must be able to handle the

many participants that may be brought in to address complex composite application problems.

Assets and Lifecycles

As with all software components, Web Services should be identified, tracked and controlled throughout their creation, testing, deployment, use and disposition. For Web Services, versioning becomes critically important – both from a control point to insure development-level services don't make their way into production environments unknowingly, as well as management of multiple production versions that may apply to different service consumers and service levels.

Release and configuration management processes will treat Web Services as just another software component, but these processes may have new control points based on the expanded requirements (such as reuse and use tracking) presented by Web Services.

Policy, Governance and Contracts

Concurrent with lifecycle management are the rules, mechanisms and controls to ensure a company's Web Services are used as planned. These include adherence to usage policy (who can create and publish a Web Service, who can use the service, security requirements, use of replication and mirrors, etc.) and definition and enforcement of service contracts (defining service levels and quality of service, enabling consumer-provider negotiations, defining service consumption rules, etc.).

For Web Services environments, such details fall under a broad umbrella of "governance." Not only do the rules, policies, guidelines have to be developed, but also the processes to implement and manage them must be in place as well as designated owners and tiers of authority.

For some organizations, much of the governance activities can occur at design and build time. Putting in place rules effecting usage and control of Web Service registries, security requirements, routing requirements and the like may work for well-disciplined development organizations. However, this will not always be the case.

IT operations organizations often find themselves receiving applications from multiple development organizations (as well as commercial packaged applications). For these organizations, the governance activities may predominantly happen at "run time." Web Services may be placed in production with no definitional context provided by a robust

registry. These so-called "rogue services" still need to be discovered, identified and tracked. Determination of Web Service policies (e.g., what assertions make up the message headers) such as encryption, routing, and so on may require a production-level "Web Service administrator." New roles will emerge to tackle the governance issues around Web Service adoption – no matter where that governance occurs.

Securing Success

As part of the overall governance model, Web Services present their own security issues. This can include encryption of Web Services traffic (i.e., XML document encryption), identity and access management for service requestors, and the implementation of XML-class firewalls and gateways. These requirements should be part of an integrated security model for the enterprise, and not treated as a security silo.

Often, existing identity and access systems are in place to support authorization and authentication integration, single-sign on, and the like. Historically, these systems group users into specific roles that map to access of particular applications or to perform sub-functions within the applications. The identities managed by these environments must have role extensions to Web Services, just as they covered discrete applications or transactions historically.

Infrastructure Underpinnings

Web Services are software components that execute on a variety of run-time infrastructures, overlaid over lower-level infrastructure elements. The availability and performance of these discrete pieces will also have an impact on service availability and performance. Whether the perspective is from a top-down model (e.g., monitor the transaction and drill down from there) or from a bottom-up model (e.g., a particular infrastructure resource is having a problem, what services does it impact?), the logical connection of a Web Service to its supporting infrastructure has to be in place to manage the "stack" integration.

While Web Services typically execute within an application server (often called a "container"), new types of infrastructure elements (and physical devices) are appearing to create a more robust and customized run-time environments for Web Services. These range from new kinds of middleware (e.g., Enterprise Service Buses or ESBs), intermediaries (e.g., eXtensible Markup Language [XML] brokers and proxies), accelerators (e.g., XML processing appliances), enhanced firewalls and routers (e.g., XML firewalls), etc. These components and devices can act as intelligent switches in the network and can

help transform data formats, perform load balancing, enhance monitoring capabilities and act as a policy or contract enforcement point.

Awareness of these new infrastructure elements is necessary, and the usual availability and performance management must be applied.

Everything Old is New Again

While the adoption of Web Services to enable new application architectures brings new wrinkles to management of these environments – organizations with robust operational process models should be able to leverage their consistency of approach inherent in process execution. In one sense, SOA and Web Services management is just “application management” repeated.

At the most base level, treating Web Services as “just another resource” to be managed is an appropriate mindset, and applying rigorous disciplines such as asset management, change management, problem management, security management, and service level management is the right way to act. While these processes may have to be modified to accommodate uniqueness exclusive to how the organization treats Web Services, the common semantics and process flows should be consistent with what the “well-managed” IT shop has today.

Where management tools have to engage and “touch” the new elements (Web Services, new types of enabling infrastructure, etc.), organizations will have to procure new tool-sets, or compatible upgrades to their existing management applications.

Conclusion

The adoption of SOAs and Web Services offers tremendous promise. Not only are these methods and technologies accelerating the “business requirements to production system” cycle, they can also streamline application integration well beyond the boundaries of a company’s IT infrastructure. By enabling high reuse rates and flexible assembly of code to support business functions, SOA and Web services hold the promise of reducing the expensive burden of system change in technology organizations.

When it comes to managing in this new world, IT organizations should seek an aggregated tool set – where the solution is integrated vertically with similar functions that are up and down the technology stack, and integrated horizontally across operational process execution boundaries. Over time, organizations are well served by ensuring that management technologies are fully integrated into the larger operational picture so that they can reap the benefits of this emerging architecture and the services delivered to the business.