



ACTIVATE BUSINESS WITH THE POWER OF I.T.™



The Well-Managed Web Service

Herb VanHook, BMC Software

Table of Contents

- ABSTRACT AND TRENDS** 1

- INTRODUCTION** 1
 - Service-Oriented Architectures..... 1
 - Web Services 1
 - Rise of the Business Service 2
 - Web Services Adoption 2

- SOA/WEB SERVICES MANAGEMENT** 3
 - Visualizing the Business 3
 - The Basics: Availability, Performance, and Throughput..... 3
 - Change, Configuration, and Governance 4
 - Asset Management Lifecycle 4
 - Policy Management 4
 - Securing Success 5
 - Metadata Management..... 5
 - Pervasive Intermediation and Business Monitoring 5

- INFRASTRUCTURE UNDERPINNINGS** 5

- EVERYTHING OLD IS NEW AGAIN** 6

- CONCLUSION** 6

Abstract and Trends

This paper discusses successful operational management of the emergent technologies represented by the adoption of Web Services for application integration. To set the stage, it is important to recognize that 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, 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 across the architecture layers for SOA-based (and non-SOA) applications, and to drive the reuse of standards-based components.
- > Increased use of programmable business process languages to describe business processes, workflows, and services — and as a macro driver for SOA implementations.
- > Integrated and 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.

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, but semantically integrated composite applications.

Introduction

Service-Oriented Architectures

Service-oriented architecture (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 a technology-independent approach to thinking about a number of very old technology-related problems.

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 (in different administrative domains and in entirely different geographies), as long as they obey the rules of the standard interface and provide the requested service.

Web Services

Web Services are a set of interface standards (message formats, protocols, etc.) to provide interoperability between software applications running on disparate platforms that are otherwise completely independent of each other and have no other knowledge or dependence between them. They could actually run on the same platform, but this is just a specific case of an otherwise generalized requirement. Web Services are used to integrate software running on different machines. The protocols and data formats are text-based where possible but this is an area of rapidly evolving technology and the focus of much recent innovation.

Web Services, along with SOA, are driving a new form of application architecture 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 from use within the World Wide Web; therefore, the term "Web" was used in the name (otherwise the word "Web" is actually more a source of confusion than clarification).

Rise of the Business Service

IT and business alignment (common goals, common vocabulary, common metrics, common IT management processes, 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 SOA principles 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 discrete 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.). In fact, new application architectures do not replace the old, but rather layer on the existing models in such a way that the new architectures are really a combinational form of everything that has gone before.

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 and .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. Whole new business models will emerge around discrete electronic, digitized business services offerings (e.g., electronic or virtual supply chains). Additionally, 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 standalone entity (it is “self governed”), 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. This critical fact makes the Web Service a discrete technology and business asset, and a viable management focal point. 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 are addressing this possibility, and IT organizations will increasingly leverage the flexibility this approach promises.

SOA/Web Services Management

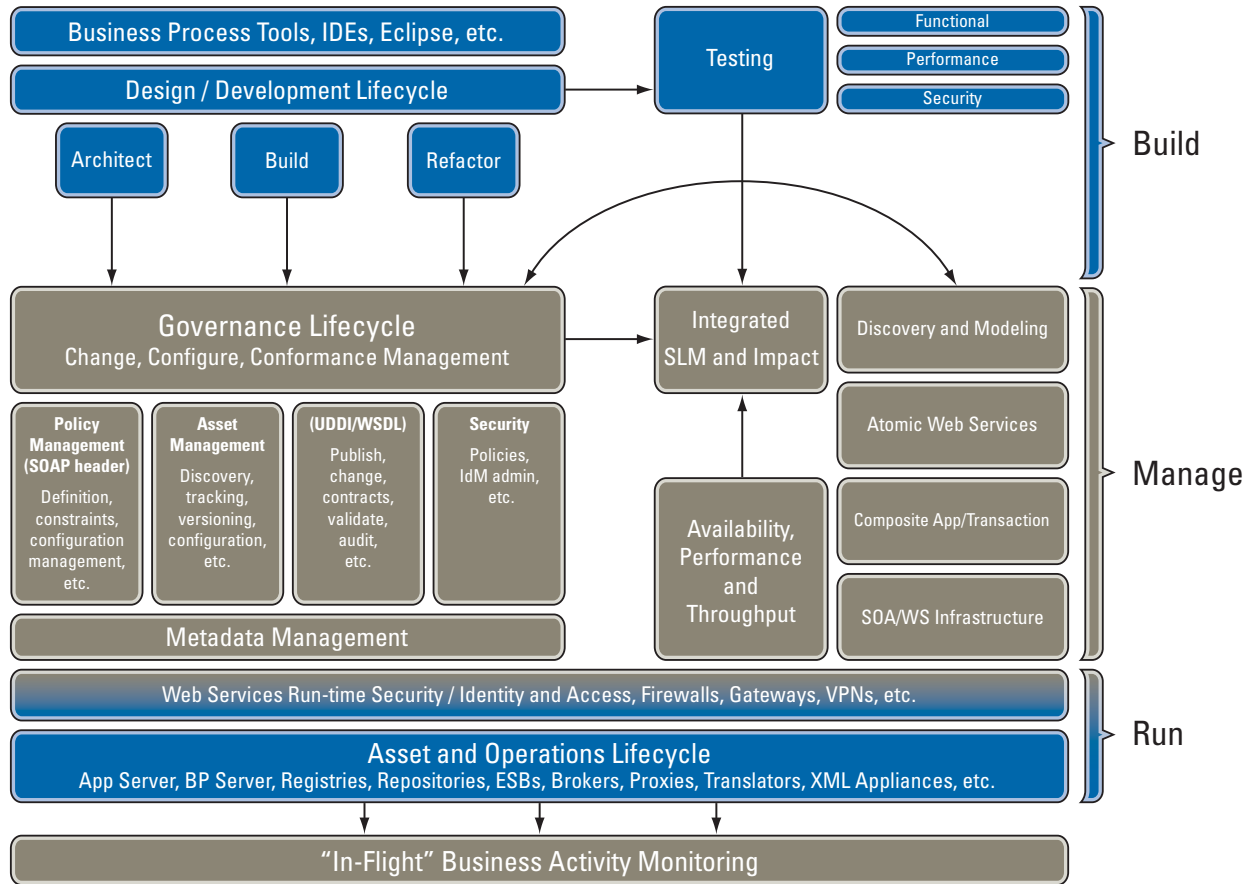


Figure 1. SOA/Web Services management functions

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. SOA introduces a new logical unit of segmentation for software systems, and obtaining insight and control over that new logical unit of segmentation is necessary for management.

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. Likewise, the more the composite application topologies provide a true proxy of the business process models (thus the elements

of the business itself), then the tools used to manage these composite applications will also become proxies for managing the business processes and the business itself.

The Basics: Availability, Performance, and 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?
- > Is it healthy?
- > Is it working well?
- > How fast is it working?
- > How much work is going through the interface?

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 locally monitoring 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.

Adoption of Web Services introduces performance concerns caused by loosely coupling software components and the use of potentially bandwidth-intensive extensible markup language (XML) documents. For performance-critical applications and transactions, organizations will consider many optimization techniques — from dedicated XML appliances to services-caching options.

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 dependency on centralized functionality represented by the 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. Auditing, logging, and metering of transaction and performance data will emerge as a management need in some organizations.

Aggregating availability, performance and throughput information and mapping it back to individual service contracts (or broader service level agreements) gives a start toward an integrated service level management (SLM) model. Additionally, organizations should strive for an overall impact management model — not only understanding the business impact of a failing or poorly performing Web Service, but also understanding the impact business demands and changes will have on their overall SOA environment.

Change, Configuration, and Governance

The traditional IT process lifecycles of change and configuration must be applied to the Web Services environment. Early adopters are treating management approaches in this area as independent functions, mainly due to the limitations of current management technology. Over time, users must integrate Web Services management into their mainstream management practices. These lifecycle models stretch across multiple management disciplines and lifecycles (below). Currently, the term “governance” is loosely applied to each of these areas, but an end-to-end process workflow model is the only way to provide a true governance picture.

Organizations must consider 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 change a service, when can it be changed, who can use the service, 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.” Additionally, validation and conformance tests appear in this area. Moving a Web Service into “production” can mean that it has to pass a series of gates to ensure it is robust, manageable, and conforms to all internal and external rules and guidelines. Not only do the rules, policies, and guidelines need to be developed, but they must be in place, as well as designated owners and tiers of authority.

Asset Management Lifecycle

As with all software components, Web Services should be identified, tracked, and controlled throughout their creation, testing, deployment, use, and disposition. Use of Web Services potentially presents a variety of assets to manage — the Web Service definition, the XML schema and documents, the Web Service execution logic (code), the Web Service policy, the service contract, etc.

Release and configuration management processes may just treat Web Services as just another software component, but these processes must have new control points based on the expanded requirements (such as discovery at development time, reuse, use tracking, publication criteria, etc.) presented by Web Services. The main goal of good software asset management in this respect is to drive up service reuse and exponentially lower development costs over time. For Web Services, versioning becomes critically important — both from a control point to ensure development-level services don’t make their way into production environments undetected, as well as management of multiple production versions that may apply to different service consumers and service levels.

Policy Management

While Web Services definition language WSDL describes what appears in the body of the SOAP message, the usage preferences called “policy” appear in the SOAP header. Strictly speaking, Web Service policies describe the capabilities and constraints of a Web Service. These typically take the form of a set of assertions (e.g., protocols supported, security required, transformations required, privacy restrictions). While these policies could be defined at design or development time, developers should only care about the business function of the Web Service. It makes more sense to decouple policies from the pure development effort and move them to the realm of Web Service administration and management, where they can be centrally and independently managed and “bound” to the Web Service closer to run time. Policy definition, versioning, updating, etc., all become administrative tasks closer to “run time,” and just like everything else in the Web Services world, the policy gets described in an XML document.

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.

Metadata Management

Use of Web Services spawns its own set of artifacts that will require management and tracking. In this case, metadata mainly includes all relevant XML schemas and documents (WSDL definitions, policy definitions, business documents, etc.). Understanding and controlling what each of these documents are, where they live, who are the owners, etc., all must be addressed.

Pervasive Intermediation and Business Monitoring

SOA designs and Web Services implementations open themselves up for intermediation. This means that messages flowing through a service infrastructure are available for inspection, auditing, policy enforcement, and transformation. This will drive a reduction in hard-coded application logic by moving the business logic and rules into the messages (XML documents) and message handling components (network devices, XML brokers, etc.). Additionally, this presents opportunities for a new type of business activity monitoring. Traditional business monitoring techniques fetched data from corporate data stores to present it as business performance indicators and statistics. By watching the content of Web Services documents as they pass through the network, a more dynamic version of business monitoring will emerge.

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 must be in place to manage the "stack" integration.

While Web Services typically execute within an application server of some form, new types of infrastructure elements (and physical devices) are appearing to create more robust and customized run-time environments for Web Services. These range from new types of middleware (e.g., enterprise service buses or ESB), intermediaries (e.g., XML brokers and proxies), accelerators (e.g., XML/XSLT processing appliances), enhanced firewalls and routers (e.g., XML firewalls), and so on. 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. This integrated service infrastructure is emerging concurrently with the adoption of SOA designs and implementations.

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. 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 need 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 toolsets, 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 toolset — 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.



ACTIVATE BUSINESS WITH THE POWER OF IT.™

About BMC Software

BMC Software helps IT organizations drive greater business value through better management of technology. Our industry-leading Business Service Management solutions ensure that everything IT does is prioritized according to business impact, so IT can proactively address business requirements to lower costs, drive revenue, and mitigate risk. BMC solutions share BMC® Atrium™ technologies to enable IT to manage across the complexity of diverse systems and processes — from mainframe to distributed, databases to applications, service to security. Founded in 1980, BMC Software has offices worldwide and fiscal 2005 revenues of more than \$1.46 billion. BMC Software. Activate your business with the power of IT. For more information, visit www.bmc.com.

About the Author

Herb VanHook is vice president of business planning at BMC, and has held several key positions at META Group (most recently serving as interim president and chief operating officer). He has more than 30 years of experience in information technology — including senior positions at IBM, Computer Associates, and Legent Corporation.

