zapthink white paper

EFFECTIVE GOVERNANCE OF THE SERVICE LIFECYCLE

REGISTRY/REPOSITORY BEST PRACTICES

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com

EFFECTIVE GOVERNANCE OF THE SERVICE LIFECYCLE

REGISTRY/REPOSITORY BEST PRACTICES

July 2008

Analyst: Jason Bloomberg

Abstract

Service-Oriented Architecture (SOA) governance, consisting of creating, communicating, and enforcing the policies that apply to the design, deployment, and management of Services, is one of the most critical, yet confusing, part of an effective SOA plan. One of the reasons why SOA governance is so important, as well as challenging, is that SOA teams must govern the entire Service lifecycle, not just as the team models and develops the Services at design time, but also as they deploy and manage them at run time, and reconfigure and recompose them at change time.

An important part of the SOA infrastructure for enabling SOA governance is the registry/repository. Today's registry/repositories are fully featured SOA metadata management solutions that coordinate policy activities and enable policy enforcement across the Service lifecycle. As organizations roll out their SOA initiatives, it is important to consider SOA governance as one of the initial steps, and a registry/repository is often the most important, first piece of SOA infrastructure to put in place.

All Contents Copyright © 2008 ZapThink, LLC. All rights reserved. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

I. The Context For SOA Governance

Few topics in today's organizations present such a diverse set of both business and technology challenges as *governance.* Governance consists of creating, communicating and enforcing the policies that are important to a company. However, what constitutes a policy and what activities and tools the organization requires are questions that have a broad diversity of answers.

Nowhere is the disconnect between interpretations of governance more pronounced than in the contrast between lines of business and information technology (IT). From the business perspective, top executives as well as government regulators set policies for the organization, which explain in often broad terms how various individuals within the company must act in certain circumstances. From the IT perspective, however, governance covers a range of policies that span the gamut from purchasing and hiring policies all the way to firewall and coding policies and enforcing service-level agreements.

Into this murky environment comes the notion of Service-Oriented Architecture (SOA) governance. On the one hand, it's important for IT shops that implement SOA to govern those initiatives, both at design time as well as run time—what ZapThink calls "SOA governance in the narrow." But even more importantly, as organizations adopt SOA, they are able to leverage the new architecture to implement better IT governance more broadly, and by extension, better corporate governance overall—"SOA governance in the broad."

Since SOA involves enterprise-wide architectural change, SOA governance should not focus solely on certain technologies or IT projects.

Since SOA involves enterprise-wide architectural change, SOA governance should not focus solely on certain technologies or IT projects. And yet, organizations should still take an iterative approach to SOA broadly, as well as with SOA governance initiatives, which generally require a focus on SOA governance in the narrow before moving onto SOA governance in the broad.

On the flip side, one way to describe SOA governance in the narrow is the structuring of decision making authority for developing and modifying various SOA artifacts. Furthermore, SOA governance encompasses people and their roles, technologies and tools, as well as processes. Knowing where to start, and how to implement an initial SOA governance framework, then, become key questions early on in any SOA initiative.

The Risks of Insufficient SOA Governance

To fully understand the importance of SOA governance, let's take a look at what happens when a SOA initiative does not have proper governance in place. First, an ungoverned SOA implementation can lead to unintended consequences, including the lack of sufficient reuse, inconsistent, duplicate, or incompatible Services, increased support costs, failure to satisfy service-level agreements, and challenges with updating the versions of Services. Furthermore, because

Thank you for reading ZapThink research! ZapThink is an IT advisory and analysis firm that provides trusted advice and critical insight into the architectural and organizational changes brought about by the movement to XML, Web Services, and Service Orientation. We provide our three target audiences of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

Earn rewards for reading ZapThink research! Visit www.zapthink.com/credit and enter the code **WSREGREP.** We'll reward you with ZapCredits that you can use to obtain free research, ZapGear, and more! For more information about ZapThink products and services, please call us at +1-781-207-0203, or drop us an email at info@zapthink.com.

regulatory compliance can be very complex, with multiple regulations and jurisdictions in play at once, and also because regulations are essentially arbitrary, lack of governance can lead to unexpected noncompliance with regulations, by failing to associate key policies with Services.

Lack of sufficient governance can also lead to various security breaches by allowing arbitrary access to Services, leading to the exposure of confidential information, unwanted access to internal systems, and other security threats. Fundamentally, without proper governance, all of the business benefits of SOA, including increased agility, reduced costs, increased asset reuse, and improved visibility, are all at risk.

In other situations, an organization has implemented SOA governance on a departmental level i.e. decentralized governance, but has not provided for governance at enterprise level i.e. centralized governance. Departmental SOA governance implementations can lead to inconsistent policies and processes, which can in turn impede an organization's ability to achieve the reuse and agility benefits that SOA promises.

It's also important to note that organizational challenges can impede a SOA governance initiative, just as they can the overall SOA effort. In fact, a poorly organized SOA governance team can become a bottleneck or even a single point of failure for a SOA initiative. Run time performance can also derail a SOA rollout. Performance is always important for production applications, and adding a Service abstraction increases the performance overhead. Run time SOA governance must address, not contribute to, such issues.

Getting Started with SOA Governance

Proper planning is important to implementing SOA governance successfully. One of your first steps should be to create your governance framework.

While SOA governance in the broad is an important benefit of SOA for an organization long term, it is essential to work though the more narrow SOA governance issues very early on in the SOA initiative. Furthermore, proper planning is important to implementing SOA governance successfully. One of your first steps should be to create your governance framework, which delineates the scope of your SOA governance efforts. That framework should then lead to the following activities:

1. *Publish the goals of your SOA initiative* – Define the business value of each phase of the SOA rollout and inform stakeholders and other interested parties about the initiative.
2. *Define the SOA organizational structure* – Put together the SOA Center of Excellence, which should help to educate the SOA governance team by fleshing out the necessary roles for the SOA initiative, including the SOA architect, Service developers, the SOA quality team, business analysts and process specialists, and operations and security personnel. Assign each member of this team a clear set of responsibilities. The SOA governance team should be a subset of this broader SOA team, but at the least the SOA governance team should include one representative from both business and IT, have the ear of upper management, and the mandate to decide SOA-related issues.
3. *Define SOA governance processes* – Consider basic issues like Service design, development, deployment, operation, and change processes. Define the roles that are relevant for the various tasks in your SOA initiative.
4. *Evaluate technical challenges for SOA governance* – Utilize SOA governance tools, including knowledge management and collaboration

Copyright © 2008, ZapThink, LLC

tools, as well as a SOA registry/repository for storing and managing SOA assets.

Organizations should also be sure to determine governance of the SOA infrastructure, including how to integrate new technologies and tools into the existing IT landscape, and how to govern processes that pertain to the various parts of the SOA infrastructure. Organizations should consider which design artifacts, interfaces, policies, and other artifacts they should store in a SOA registry/repository.

II. Critical Need for Governing the Service Lifecycle

One of the most important considerations you should have when fleshing out your governance framework is the fact that governance applies across a range of SOA activities. SOA governance applies during design time, as developers, Service specialists, and process specialists plan and implement Services. It also applies at run time, when your operations personnel manage running Services. Furthermore, governance also applies during change time, as your SOA team reconfigures and recomposes Services to meet changing process requirements. We call this span from design time to run time to change time as the Service lifecycle. In particular, the design time and run time aspects are very different parts of the Service lifecycle, and often have divergent overlapping needs, as decisions organizations make during design time directly influence run time results.

The design time portion of the Service lifecycle spans Service planning, design, discovery, implementation and testing phases, while the run time aspect encompasses deployment, operations, support, and versioning tasks. Change time governance is in some ways an aspect of run time governance, but is fundamentally is a separate phase in the SOA context. The change time portion of the Service lifecycle includes changes to configurations and compositions of running Services. Such changes need not impact the underlying execution of the Services, and yet managing and enforcing policies during this change time phase is an especially important part of the SOA governance picture.

To better delineate these Service lifecycle phases, the lists below illustrate some of the activities various personnel might undertake in each phase:

Design time:

- A business analyst gathers requirements for new or changed Services
- An architect designs a Service contract or plans a Service implementation
- An architect or business process specialist models and specifies business processes
- A developer searches for an existing Service before building a new one
- A developer requests that the SOA Center of Excellence approve the creation of a new Service
- A developer creates new Services or assembles existing Services
- A tester simulates a new Service to execute a test plan.
- An architect crafts a Service domain

Copyright © 2008, ZapThink, LLC

Run time:

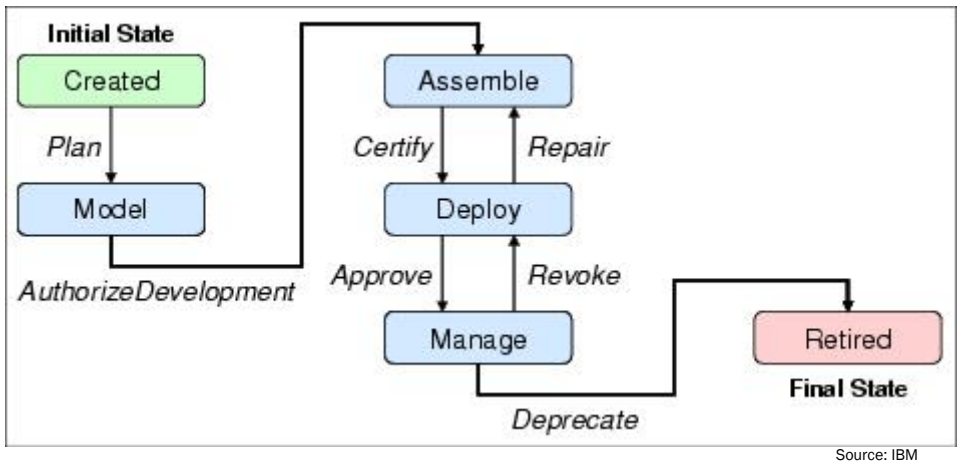
- Architects and developers compose Services to meet new process requirements.
- The operations team places Services into operational environments
- The operations team manages Services to ensure compliance with business policies
- Operations personnel monitor Services to facilitate an understanding of how well the business or IT systems are performing
- A business executive monitors a business process
- A manager confirms compliance of a Service with particular policies or service level agreements
- The CIO monitors levels of Service reuse
- The operations manager handles Service exceptions
- Service consumers are able to dynamically discover and bind to Services and associated metadata.

Change time:

- A business analyst plans a change within a certain business process
- The SOA team reassembles and redeploys changes to Service configurations and compositions to meet changing process needs
- An IT administrator adjusts the quality of service requirements for a Service.
- Data analysts modify data and semantic models
- Architects and developers alter Service-related metadata

A simplified illustration of the Service lifecycle is in the figure below:

The Service Lifecycle



Source: IBM

Governance of the Service lifecycle essentially combines the design time, run time, and change time phases with the SOA governance framework.

The goals of governing the Service lifecycle include defining who in the organization has the relevant authority to make changes, providing transparency into the effects of those changes, defining and enforcing rules for Service creation, use, and management, managing change, measuring results, and optimizing Service behavior. Governance of the Service lifecycle essentially combines the design time, run time, and change time phases with the SOA governance framework. Basically, for each of the phases, the steps are to determine which policies are within scope, and then delineate how you will create, communicate, and enforce those policies in each phase, as well as obtain visibility into levels of compliance.

The full breadth of the Service lifecycle involves Services, related artifacts, and roles. In practice, a business process analysis and optimization request could result in either the creation of a new Service, modification of an existing Service, or reuse of an existing Service as-is. SOA governance provides guidance by describing how new Services move from planning and design to production, mandating the consideration of Service reuse or modification as appropriate, ensuring that necessary reviews are an integral part of each phase, and defining each person's role within the Service lifecycle.

Iterative SOA Governance

It's critical to incorporate some aspects of SOA governance into the earliest stages of SOA planning.

In order to minimize problems as the organization increases the scope of its SOA governance efforts, it's important to tackle the implementation of SOA governance iteratively, as an integral part of an overall iterative approach to the SOA rollout. As a result, it's critical to incorporate some aspects of SOA governance into the earliest stages of SOA planning. At the least, initial SOA governance iterations should include defining and publishing your SOA goals, defining the SOA governance organizational structure, and sketching out initial SOA governance processes.

Among the advantages of taking an iterative approach is that because you're starting with a small project, communication is straightforward because the team is small, so the scope of the governance decisions are limited. You don't yet have to worry about a central repository for Service artifacts. As you expand your deployment, however, those governance issues increase and become increasingly urgent because now you've expanded out across a larger organization.

Early iterations of an organization's SOA Governance approach should include the evaluation of the technical issues that governance raises, and often the purchase of a registry/repository. Ensure that such tools will scale through future iterations. It also makes sense for your architect team to formulate a standard approach to representing policies, in order to facilitate interoperability into the future.

Next, organizations should focus on how to optimize their lifecycle processes, as well as the various SOA artifacts that will continue to change over time. To support this change, it's essential to implement flexible integration with different metadata stores. Furthermore, organizations should realize that policies themselves have a lifecycle as well. Be sure to hammer out how to create, implement, version, and deprecate policies just as you would the Services that they apply to.

Copyright © 2008, ZapThink, LLC

III. The Role of an Enterprise Registry/Repository: More than Governing the Service Lifecycle

Even though governance is clearly about far more than technology, nevertheless, the registry/repository is the key infrastructure requirement at the center of the SOA governance value proposition. Therefore, understanding what a registry/repository is, why it's important to SOA governance, and also what value a registry/repository provides beyond SOA governance are critical elements of your SOA roadmap. And to understand how registry/repositories came to be at the center of this storm, it's important to understand the role of *metadata* in SOA.

Narrowly defined, metadata are data about data, but more broadly, metadata are information about the workings of a system, as opposed to the information, or data the system works with. In the SOA context, important metadata include Service contracts, BPEL files and other process-centric artifacts, configuration files, test plans, schemas, policies, and more.

It is the fact that we can represent many policies as metadata that is one of the primary reasons that registry/repositories are so critical to SOA governance. If we have a way of representing certain policies in a standard format, say as an XML file that might conform to WS-Policy or other standards, then we can use a tool like a registry/repository to store, communicate, and enforce the policy, as well as provide workflows for dealing with such policies across the Service lifecycle.

It's not practical to represent all kinds of policies as metadata, however. Many design time policies, such as the policies for discovering, creating, publishing, and reusing Services, are typically for human, rather than computer, understanding, and therefore, metadata can't easily represent such policy information. For those aspects of policy that lend themselves to representation as metadata, registry/repositories are important tools for managing and enforcing such policies as well.

As a practical example of how a registry/repository works, envision that a developer is trying to reuse a Service that another developer, the owner of the Service, had published in the registry/repository previously. The owner of the Service then decides to change the Service. Before the developer can change it and publicize the change, the registry/repository sends a notification so that other developers can know there's a proposed change in the works so that the team can evaluate what effects that change will have.

This design time example is but one instance of the power and usefulness of a registry/repository. Fundamentally, SOA has the potential to drive business flexibility, performance, and innovation by aligning technology with business objectives, but in order to accomplish this admittedly difficult feat it is essential that you manage the services in your SOA implementation throughout the lifecycle with a registry/repository. Also, remember to work through registry/repository deployment considerations, including scalability, reliability and high availability of the solution, efficient and high performance access to the content, and sophisticated lifecycle management support, including promoting services from test to production. In this way you will be able to scale your registry/repository solution regardless of how extensive your SOA implementation becomes in the future.

Copyright © 2008, ZapThink, LLC

Governance of Service Metadata

The Service metadata in the registry/repository enables SOA teams to select, invoke, govern, and reuse Services. Registry/repositories store information about Services in your environment, or in other organizations' environments, that you already use, that you plan to use, or that you want to be aware of. For example, an application can check with the registry/repository just before it invokes a Service to locate the Service that best satisfies its functional and performance needs. This capability helps make your SOA deployment more dynamic and more adaptable to changing business conditions.

Registry/repositories typically include a Service registry that contains references to specific information about Services, such as the Service interfaces, its operations, and parameters found in the Service contracts, as well as a metadata repository that stores and manages a wide variety of SOA-related artifacts. Registry/repositories often support the viewing of relationships between Services and related metadata, enabling SOA team members to visualize the complex relationships between Services and metadata and other artifacts the team has stored in the registry/repository. Such a visual representation of the metadata makes it easier to understand the impact of changes in the SOA environment.

Registry/repositories enable SOA team members, as well as Service consumers, to search for the Services they require for a particular task, and then to obtain the information they need to bind to the Service.

Service discovery capabilities are also an important part of the role of the registry/repository. Registry/repositories enable SOA team members, as well as Service consumers, to search for the Services they require for a particular task, and then to obtain the information they need to bind to, or use the Service. To support this search and discovery capability, most registry/repositories support the Universal Description, Discovery, and Integration (UDDI) Web Services standard. However, UDDI support is only a small part of the capabilities a modern registry/repository offers, and many advanced products offer multiple different approaches for accessing Service metadata stored in a registry.

Because registry/repositories provide visibility into Service associations and relationships, they provide control over the dynamic SOA environment. Furthermore, registry/repositories can take advantage of Service interoperability, primarily through standards support. This interoperability can also help unlock the value of mainframe applications, databases, and other legacy capabilities. Once the SOA team exposes mainframe applications and database interactions as Services, it's possible to publish those Services in the registry/repository for future access by both mainframe and non-mainframe applications without the need for further programming.

Scaling a registry/ repository solution also helps teams iteratively adopt SOA, as they move through the Service lifecycle activities of modeling, Service assembly, deployment, and management.

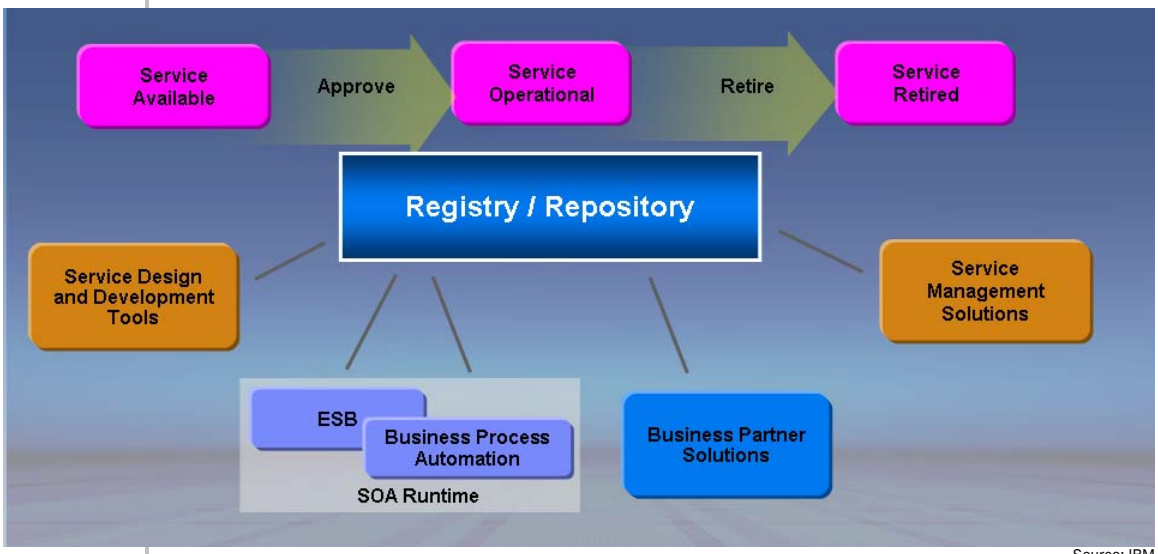
Scaling a registry/repository solution also helps teams iteratively adopt SOA, as they move through the Service lifecycle activities of modeling, Service assembly, deployment, and management. During service modeling, teams can use the registry/repository to create or reuse service taxonomies, classifications, and XML schemas. During Service development or assembly, they can use it to locate Services for reuse and to enable Service composition. Then, as part of the Service deployment process, teams can publish Service descriptions to the registry/repository, which can then augment any Service descriptions that already exist. The registry/repository integrated with service management solutions can capture and assess the performance of Services against business and operational performance objectives in order to manage Services that leverage system management best practices such as those in the Information Technology Infrastructure Library (ITIL).

Copyright © 2008, ZapThink, LLC

Managing the Service Lifecycle

Registry/repositories, interact with the infrastructure that supports specific phases of the Service lifecycle and capture more detailed information about Services relevant in each phase of the lifecycle, as shown in the figure below:

Registry/Repository across the Service Lifecycle



Source: IBM

During the design time, model and assemble phases of the Service lifecycle, the SOA team uses the registry/repository to locate the copies of record of candidate Service metadata as well as policies governing the interactions with those Services. They can also publish and govern Service metadata about emerging, to-be-deployed Services. In essence, during design time, the registry/repository governs the gathering of requirements, modeling, design, discovery, construction, testing, and composition steps.

At run time, the core challenges are integrating people, process, and information assets as well as managing applications, Services, identity, compliance, and business metrics. Registry/repositories provide the system of record for metadata describing Service interaction endpoints. SOA run time environments like ESBs, application servers, and intermediaries access relevant metadata in the registry/repository in support of both deployer and administrator roles to drive operations of deployed composite applications as well as individual Services.

Once the development team has finished their work and completed testing, deployers add further to the Service metadata, providing binding information for service endpoints that composite applications use and managing deployment of the metadata from the development environment to the staging and production instances of the registry/repository. The production registry/repository is then able to share Service metadata if policies in place allow. Relevant metadata are also available to the run time systems and those user roles that are responsible for the management of those systems.

Various parts of the IT infrastructure can also dynamically retrieve policies from the registry/repository that are in effect for a Service interaction for the purposes of logging, filtering, data transformation, or routing. Such automated policies

Copyright © 2008, ZapThink, LLC

have a high level of enforcement, depending upon the policy management infrastructure in place. Enforcement of a security policy, for example, should ideally offer no alternatives to compliance—in other words, the infrastructure should prevent undesirable behavior.

We refer to this level of policy enforcement as run time governance, because the governance infrastructure must take a direct, active role in ensuring the compliance with particular policies during run time. Some examples of active governance include enforcing limitations on the number of Services in a composition, enforcing specific Service reuse metrics, as well as most security policies, as well as maintaining business/IT alignment as business requirements change, including financial transparency and process control.

Policy Management, Visibility, and Flexibility

Because Services abstract the underlying complexity of the technology, changes to business processes or Services can place unexpected or excessive demand on the underlying information systems.

SOA is especially useful in dynamic, heterogeneous environments, and can increase business agility in such environments. However with this increased dynamism comes additional risks, for example, the risk that someone will change a business process in a way that is detrimental to the business. Because Services abstract the underlying complexity of the technology, changes to business processes or Services can place unexpected or excessive demand on the capacity of the underlying information systems, either crashing the system or having an adverse affect on the other processes that the system also supports.

SOA also exacerbates the risk that someone will introduce rogue software, or that someone will change the configuration of the system in way that disrupts the business. For these reasons as well as others, it's important to place the Service lifecycle on a backdrop of governance processes that ensure the enforcement of compliance and operational policies, so that change occurs in a controlled fashion and with appropriate authority.

The SOA governance challenge, therefore, boils down to how to maintain adequate control while at the same time providing the flexibility the organization requires from their SOA initiative. To this end, SOA governance requires that organizations take business policies, typically in written form, and transform them into metadata-based rules that can help automate the process of validating and enforcing compliance with those policies in both design time and run time environments. Companies must then manage policies through their entire lifecycle.

At design time, the goal of SOA policy management is to detect and resolve quality issues before putting the Services into production. At run time, organizations must also implement run time policy management for monitoring and automatically enforcing policies during the usage of Services.

In general, policy lifecycle management within SOA focuses on ensuring the quality, performance and applicability of available Services, enabling Service consumers to discover and reuse Services as well as other artifacts, managing Service versions, handling the security of Services and other SOA artifacts, and assessing and managing the impact of change across all Service consumers. Managing policies also includes providing visibility into whether people are following policies, as well as handling policy infractions. Such policy management tasks are also an inherent aspect of IT governance, as well.

At design time, the goal of SOA policy management is to detect and resolve quality issues before putting the Services into production. At run time, then, organizations must also implement run time policy management for monitoring and automatically enforcing policies during the usage of Services. Such run time policies may focus on security, QoS, or other requirements for the behavior of the Services in a production environment.

SOA governance solutions require the following capabilities:

- *Policy Authoring* – defining and maintaining reusable policies over the course of the Service lifecycle. Moving proprietary policies from siloed

Copyright © 2008, ZapThink, LLC

systems to central policy management that can interface with the SOA infrastructure in a standards-based way.

- *Policy Association* – applying policies to Services and other artifacts, often through the use of a registry/repository. It's usually preferable to publish such policies to the registry/repository similar to Service contracts.
- *Policy Enforcement* – Enforce SOA policies in practice, either via the registry/repository, via SOA management tools, or via various types of policy enforcement points on the network, depending upon the type of policy.
- *Policy Reporting* – providing visibility into policy compliance via reports that the registry/repository can store or generate.

The first step for automating policy activities is to conduct a policy inventory to uncover the policies that are a priority. Next, the organization should decide which policy activities are automatable. In other words, identify those policies that you can represent as metadata that your policy management tools can understand. Then, users decide on level of granularity for those policies. Note that not every policy management or enforcement tool represents policies with the same level of detail, so it's important to develop a consistent format for representing the policies.

At this point you must translate policies into a system-understandable format. Standards like WS-Policy and WS-SecurityPolicy can aid somewhat with this formatting issue, but unfortunately, these standards can only help in rather narrow situations. In the general case, it will be important to either develop your own XML-based policy specification, or encode the policies directly into the policy enforcement system, which represents policies according to the tools' own internal specifications.

Once you have fully defined your policies, you must figure out how to enforce policies in practice. Policy enforcement essentially depends on the type of policy. For example, an XML firewall might enforce a security policy, while a registry/repository might enforce a Service reuse policy. SOA management tools enforce many run time policies, while identity and access control solutions are adept at enforcing access management policies. Finally, it's important to identify techniques for long-term policy maintenance, as the organization creates, modifies, and retires its policies.

IV. IBM WebSphere Service Registry and Repository: Broad-Based, Scalable SOA Governance

The IBM WebSphere Service Registry and Repository (WSRR) is an increasingly significant player in the SOA registry/repository market. WSRR is one of the leading SOA registry/repository offerings on the market. It is an enterprise-scale registry/repository with highly available and reliable infrastructure. WSRR is an optimized Service metadata repository with an extensible model and classification approach. It stores and manages Service metadata for ESB and BPM solutions with high performance caching mechanisms. WSRR offers Service lifecycle management, metadata federation, SOA governance and policy management on a reliable, high-performance platform that leverages the WebSphere Application Server, and integrates with over twenty other products and solutions. It is further extensible using open standards as well.

Copyright © 2008, ZapThink, LLC

WSRR enables organizations to bring together Service metadata that are otherwise scattered across an enterprise to provide a single, comprehensive description of a Service.

As the integration and collection point for Service metadata, WSRR establishes a central point for finding and managing Service metadata from a number of sources, including Service deployments and other and other endpoint registries and repositories. WSRR enables organizations to bring together Service metadata that are otherwise scattered across an enterprise to provide a single, comprehensive description of a Service. WSRR is then able to increase visibility of Services, manage versions of Services, and then analyze and communicate proposed changes to those Services. WSRR also integrates with Service monitoring tools to capture metadata about performance and availability that other parts of the SOA infrastructure can access as the copy of record of those metadata.

WSRR provides awareness of Service associations and relationships while encouraging reuse of Services to avoid duplication and reduce costs. It also enhances connectivity with dynamic Service selection and binding at run time and enables governance of Services throughout the Service lifecycle. Furthermore, it ensures interoperability of Services via open standards, and also leverages such standards to interoperate with other standards-based registry/repositories to provide a unified view across a variety of Service information sources.

WSRR promotes Service reuse and eliminates redundancies by enabling SOA teams to publish and find Services and related metadata, as well as to create new business processes using those Services. WSRR also provides run time capabilities, including optimized access to Service metadata for managing Service interactions and policies and enhancing Services with rich Service metadata.

WSRR also offers SOA policy management capabilities, and enables consistent enforcement of policies across the SOA implementation. WSRR treats policies as metadata that it can store, classify, change and govern just like Services. WSRR's policy authoring editors allow team members to create new policies and visualize the WS-Policy structure ensuring that new policies are consistent. WSRR also supports policy libraries for creating collections of policies.

WSRR also provides certain policies out of the box for capturing best practices for governing the Service lifecycle, as well as operational policies that provide common templates that encourage interoperability. WSRR can serve as a core part of the SOA policy management infrastructure that includes policy enforcement points such as ESBs, application and Web servers, network devices such as security and XML appliances, complex event processing products and security management tools.

Finally, WSRR enables better control of SOA with governance by facilitating the Service lifecycle by analyzing the impact of Service introduction, deletion, or alteration. It also manages role-based access to Services, as well as Service versioning and retirement.

Each phase of the Service lifecycle has different challenges requiring targeted repositories. In order to consistently govern the SOA across all phases of the lifecycle, and yet cater to different user needs in each phase, IBM's strategy is to build optimized SOA repositories that federate Service metadata for governing the Service lifecycle end-to-end.

To that end, WSRR's Advanced Lifecycle Edition, which consists of integrated design time and run time repositories to govern the Service lifecycle from creation to consumption. At the heart of this offering is Rational Asset Manager, which manages information useful for developing, reusing and managing all

Copyright © 2008, ZapThink, LLC

types of assets, in addition to WSRR, which manages information useful for the runtime operation, management, development, and use of Services.

V. The ZapThink Take

For any SOA implementation to provide the benefits of business agility in the face of heterogeneity and change, organizations require an expanded scope of governance that reconciles multiple points of view, such as business versus IT and development versus operations, and across different lines of business with different priorities. Coming up with consistent Service and process models to meet these reconciled objectives is also essential. A registry/repository like the IBM WebSphere Registry and Repository is an essential tool for achieving these objectives.

The registry/ repository supports governance of Service metadata throughout the lifecycle of a Service.

The registry/repository supports governance of Service metadata throughout the lifecycle of a Service from its initial publication in the development space to deployment to Service management. SOA governance tasks serve the entire Service lifecycle, insuring Service cooperation, Service technology compatibility, and proper Service investment and reuse.

The registry/repository also allows users to both store and register standards-based Service metadata including WSDL files, schemas and policies. The publication of Service metadata allows users to socialize their high value or standardized shared Services, thus encouraging interoperability and reuse.

The ability to query and find Service metadata at design time as well as run time supports the reuse of interfaces and schemas as well as binding of Service endpoints. The capture of Service dependencies in the registry/repository allows for the assessment of the impact of changes. It can then notify consumers of Services that such changes impact, helping users take advantage of improvements and ensure that no degradation in quality of service occurs.

WSRR provides a mechanism for associating policies to Services, which allows the infrastructure or application code to interpret and enforce such policies. The organization is thus able to achieve an agility benefit, since it's possible to configure policies in WSRR so that teams can change them quickly without redeployment or coding in a way that is still subject to governance.

Copyright © 2008, ZapThink, LLC

Copyright, Trademark Notice, -and Statement of Opinion

All Contents Copyright © 2008 ZapThink, LLC. All rights reserved. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

About ZapThink, LLC

ZapThink is an Enterprise Architecture (EA) strategy advisory firm. As a recognized authority and master of Service-Oriented Architecture (SOA) and EA, ZapThink provides its audience of IT practitioners, consultants, and technology vendors with practical advice, guidance, education, and mentorship solutions that assist companies in leveraging SOA to meet their business needs and presenting viable SOA solutions to the market. We provide this audience a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink provides IT practitioners strategic insight and practical guidance for addressing critical agility and change management issues leveraging the latest EA and SOA best practices. ZapThink helps these customers put EA and SOA into practice in a rational, well-paced, and best practices-driven manner and helps to validate or recover architecture initiatives that may be heading down an unknown or incorrect path. ZapThink assists with solution vendor, technology, and consultant selection based on in-depth, objective evaluation of the capabilities, strengths, and applicability of the solutions to meet customer needs as they relate to EA initiatives and as they map against emerging best practices. ZapThink enhances its customer's skills by providing education, credentialing, and training to EAs to develop their skills as architects.

ZapThink helps to augment consulting firms' EA offerings and intellectual property by providing guidance on emerging best practices and access to information that supports those practices. ZapThink provides frameworks for product-based consulting based on ZapThink insight and research, such as SOA Implementation Roadmap guidance, Governance Framework development, and SOA Assessments, and provides a means to endorse and validate consulting firm offerings. ZapThink also accelerates consulting firms' efforts to attract, retain, and enhance the skills of EA and SOA talent by providing education and skills development

For solutions vendors, ZapThink provides retained advisory for guidance on product strategy, as well as marketing, visibility, and third-party endorsement benefits through its marketing activities, lead generation activities, and subscription services. ZapThink enables vendors to leverage ZapThink knowledge to transform their offerings in a cost-effective manner.

ZapThink's Managing Partners are widely regarded as the "go to advisors" and leading experts on SOA, EA, and Enterprise Web 2.0 by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted experts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Baltimore, Maryland. Its customers include Global 1000 firms and government organizations, as well as many emerging businesses. Its Managing Partners have worked at such firms as IDC, Saga Software, Mercator Software, marchFIRST, and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, and ebXML.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how SOA will impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Road
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (815) 301 3171
info@zapthink.com

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com