When approaching a topic such as policy in the context of a service-oriented architecture (SOA) discussion, there is a tendency to start at the top and work down. While having a good sense of the top-level perspective on policy is essential, it is often not practical to wait until the entire realm of policies has been defined before introducing selected aspects of policy management in an SOA solution.

Often, the best approach in a new discipline is to set a general direction and then begin to build something concrete in order to test the theory and the mechanics of the execution. This provides an opportunity to refine the objectives from a practical vantage point.

In this IBM® Redpaper publication, we follow this approach, exploring some aspects of overall policy management that companies can use today. We also illustrate a few pragmatic scenarios that demonstrate techniques for enhancing an SOA solution with policy management.
Introduction to policies and related concepts

Policy management plays a key role in enabling governance in any service-oriented environment. SOA practices help businesses identify and focus on the key services of the business. By adding policies, we add points of control and agility for business and IT, making SOA more consumable, and therefore, accelerating the adoption of SOA solutions. IBM has developed a federated approach to policy management that coordinates the authoring, transformation, enforcement, and monitoring of policies across a broad range of products and technologies. With this approach, policies can be added to your SOA environment at various stages in an SOA transformation.

This paper serves both as an introduction to the concepts of policy management and as a description of an implementation of policy federation across a subset of the IBM SOA Foundation products.

What a policy is

At the highest level of abstraction, a policy is simply a statement of a specific business requirement. At this level, the policy is normally expressed in natural language (for example, English, French, and so on) and is used to communicate business requirements between business analysts and IT professionals. As the two parties begin to collaborate, this business policy is then decomposed into a set of objectives, strategies, and tactics that define the details of how the business requirement is going to be implemented and enforced across the organization.

The decomposition of the initial business requirement into a set of more detailed policies is organized into two distinct dimensions:

- A horizontal dimension (policy domains and types)
- A vertical dimension (levels of abstraction of policies)

For example, the initial requirement “protect private information” is associated, at the business level, with different areas of the business (purchase order, billing) and roles with different domains of expertise (chief security officer, chief information officer). The initial business policy is then decomposed into related policies of different types at the architectural level (for example, compliance with coding standards, checkpoints in the life cycle of selected services, and definition of rules for access control and message protection). These related policies are then further decomposed into more detailed policies, at the operational level (for example, formats for encryption, credential propagation, and definition of filters and alerts).

At each level of abstraction, a policy is associated with one or more elements of the architectural framework (referred to as subjects). For example, our “protect private information” business requirement can be decomposed into the following policies:

- Information security policies that are associated with one or more business processes at the business level
- Service lifecycle policies that are associated with service lifecycle transitions at the architectural level
- Message security policies that are associated with a service message at the operation level

The resulting policy tree is organized into progressively refined and actionable policy expressions that are classified into different domains. By using this representation, we can trace each policy back to its initial business requirement.
Why policies

The adoption of SOA best practices has created an ideal environment for the introduction of several aspects of policy management inside an SOA solution. Such concepts as service, service composition, and service orchestration are now seamlessly recognized as “first class citizens” by development tools, middleware, and management solutions. The synergy between SOA and policy management is achieved by several benefits resulting from the introduction of policies:

► Formalization

Many policy domains have been identified and formalized into a standard representation of policy expressions. The resulting reduction of the ambiguity of the characterized policies makes them readily available for automation.

► Standardization

Common policy expressions and semantics for a specific policy domain can be unified into an implementation-independent standard that guarantees that two different systems are compliant with the same policy regardless of their specific implementation.

► Automation

The formalization and standardization of policy domains, together with the recognition of these standards by tools, middleware, and management systems, allows the automatic and transparent configuration of these systems and the automatic enforcement of related policies.

► Traceability

In a federated policy management system, traceability between related policies at different levels of abstraction can be achieved by leveraging formalization, standardization, and automation. An important component of traceability is a policy information exchange mechanism between the different parts of the enterprise architecture in which policies are defined, transformed, and enforced. Formalization and standardization can provide a good basis for traceability. However, any manual management of the relationships across the horizontal and vertical policy dimensions is difficult to scale up to an enterprise level, which is where automation can play an important role.

► Reuse

Reuse can be achieved at various levels by collecting policy expressions and related values from best practices in the specific domain into a library of reusable policies. Alternatively, reuse can be achieved at a higher level, in which a tree of related policies of a different type and level of abstraction can guarantee compliance with mandates or standards at the business level.

Policy and governance

An iterative approach to governance usually addresses the most operational, easy-to-measure governance aspects first, those aspects that can be easily associated with one or more policies that can be automatically enforced. The progressive delegation to automation of selected governance and management aspects is definitely a positive trend that reduces the complexity of our SOA solution. However, as we address further and more strategic governance aspects, we realize that a more comprehensive and prescriptive approach is required for the following purposes:

► Identifying relevant policies starting from our business goals
► Defining new roles and responsibilities related to the planning, definition, enablement, and measurement of the governance aspects
Documenting the manual steps that are needed to enforce those governance aspects that cannot be automated

- Associating policies to relevant metrics in order to measure the effectiveness of the related governance aspect and its improvement
- Associating policies with specific activities and artifacts and their coordination with the manual steps

IBM has a comprehensive approach to SOA Governance that begins with the definition of the governance model and continues through to the enforcement of selected governance aspects across the service life cycle. The IBM SOA Governance and Management Method (SGMM) provides a prescriptive and iterative approach to identifying the processes, roles, responsibilities, policies, metrics, and checkpoints that compose the governance framework (Figure 1).

![Governance framework](image)

**Figure 1 Governance framework**

### Levels of abstraction for policies

As mentioned previously, managing policy across an organization can be challenging because business requirements that need to be expressed can span numerous domains. In some cases, each domain specifies its principles and roles, in different ways. Depending on the maturity of the SOA infrastructure, different policies must be expressed with varying degrees of detail, in different formats. Some are expressed by using plain English, and others are expressed by using common formats.

To navigate the policy pain points of its customers, IBM has proposed to categorize policies in three levels of abstraction (Figure 2 on page 5):

- Business
- Architecture
- Operations

The goal for any organization is to optimize its policy expressions and to use SOA principles to encourage reuse and simplicity of policy management. This optimization can occur only when businesses start to declare policies and then manage policy expressions like other aspects of their SOA practice, looking for common formats and control points.
The scenarios described in this paper show practical examples of policy enablement today that can be exploited and extended by customers who want to begin a policy enablement effort. In the following sections, we delve into examples that illustrate the architectural and operational levels of abstraction and examples of their related policies:

- **Architecture**
  Architectural policies are captured as architectural or development practices. Examples of architectural policies include compliance with reference architecture or software asset reuse policies.

- **Operations**
  Operational policies are rendered as settings and entitlements. Examples of operational policies include message security or access control policies.

**Note**: The example policies shown in this paper belong to the architecture and operations level only. We do not cover business policies.

### The life cycle of a policy

Implicit in policy as an enabler of governance is the assumption that a policy that has been authored can be enforced and that there are measures for determining whether a policy is being adhered to. Without this assumption, a business objective remains a statement of a requirement.

To understand how a policy expression can be used as a component of a policy-enabled SOA solution, we have to look more closely at the stages that a policy goes through on its way from authoring to enforcement. We refer to this as the *policy life cycle*, which is illustrated in Figure 3 on page 6.

Policies go through this life cycle starting with the requirements that are captured in a particular format or policy expression. We call this task *authoring*. Additional tasks can be included in the life cycle. For example, a policy expressed in a human readable format may (optionally) be “transformed” into another expression that can be automatically enforced by computers.
The assumption for the enforcement task in the policy life cycle is that the enforcement points are capable of understanding the expression and making sure that the policy is respected and adhered to. In the center of Figure 3, you see the different “types” of policy, which include access control, service life cycle, and so on.

This illustration is important because another aspect of the policy life cycle is the domain of the policy expression. When we note the range of policy types, we also note that enforcement points for the domains can vary from human to automated enforcement points. For example, when we look at an IT topology in which there is a security gateway, we can identify an automated operational enforcement point to enforce an “access check” on the Verify Credit service. This type of automation is possible because the gateway understands what an access check is and what the Verify Credit service is.

Monitoring is the task in the policy life cycle that collects metrics or statistics to support the process of checking for policy compliance. Monitoring is an optional part of the policy life cycle, but it is typically a key component of a governance framework. For example, our security gateway can act as a monitoring point in addition to an enforcement point and collect metrics on service access. This data can be used to correlate how well the access control security policy for the Verify Credit service is performing. For example, it can determine who was granted access and when and who was not granted access. Good monitoring also allows review of the effectiveness of a solution. After observing the behavior for an initial policy definition, the policy may be modified to better align with the objectives of the business.

Policy management scenarios

In the following pages, we describe the implementation of policy federation by using a subset of the IBM SOA Foundation products. To best illustrate the various concepts that we introduced in the first part of this paper, we have developed a sample implementation that supports four different policy management scenarios (Figure 4 on page 7). Each scenario focuses on key aspects of the IBM federated approach to policy management. Then we describe how different roles, in different phases of the service life cycle, can author, transform, enforce, and monitor selected policies. The four scenarios address issues that can arise from an SOA solution that does not adequately handle several aspects of security and governance.

To establish the context for the scenarios, we refer to JKHL Enterprises, a company that is well into its SOA transformation initiative and wants to reduce the cost of its account opening process by automating tasks and maximizing reuse. The company’s Verify Credit service is
being successfully reused by internal applications, such as Open Account, and by external agencies, such as PartnerCo.

The four scenarios can be summarized as follows:

- **Service lifecycle policy**, **Web Services Interoperability (WS-I) compliance**, and **certificate of originality policies**
  
  JKHL Enterprises wants to ensure standards compliance for its service descriptions and to verify that the code assets produced during development do not contain parts of undocumented origin or parts subject to usage restrictions. We associate one or more validations at specific transitions in the service life cycle to ensure that services do not exit development until these build-time policies are enforced.

- **Message security policy**
  
  JKHL Enterprises specifies a message security policy to prevent unauthorized access to message content, checking messages for integrity and confidentiality and ensuring that the originator of messages is known.

- **Access control policy**
  
  JKHL Enterprises uses a policy to establish detailed rules for the identification of authorized users, resources, actions, and environment.

- **Reliable messaging policy**
  
  JKHL Enterprises wants to ensure message delivery when a potential loss of messages is experienced due to failure of a component.
The scenarios involve the following human roles:

- **Development lead**
  The development lead is responsible for implementation (code). This person leads the team of developers who write the code and unit test.

- **SOA architect**
  The SOA architect leads the development of the solution (from a technical standpoint), from requirements management to deployment to production.

- **Security architect**
  The security architect leads the development of the security architecture for the solution, making sure security policies are addressed and enforced.

- **IT operations**
  The IT operations person coordinates the activities that are related to the production environment, for example provisioning, maintenance, and monitoring.

### A federated policy management architecture

Figure 5 shows the high-level architecture that is used to realize the four scenarios with the following IBM SOA Foundation products:

- Rational® Asset Manager (a reusable software asset repository)
- Tivoli® Security Policy Manager
- WebSphere® Service Registry and Repository
- WebSphere DataPower® XI 50 SOA Appliance (used as a security gateway)

---

**Figure 5  IBM products used in the federated policy management scenarios**
WS-I compliance and certificate of originality policies

This scenario addresses build-time issues, such as the following examples:

- Service descriptions do not follow standards for interoperability.

  JKHL Enterprises has established SOA principles, including separating service interfaces from service implementations, using the Web Services Definition Language (WSDL) to fully describe services for developers and consumers. JKHL Enterprises also started to publish these service descriptions to their service registry and repository. They realized, however, that not all developers from the partner companies can use the service definitions. For example, the PartnerCo agency implements its services on .NET, and the original JKHL Enterprises service definitions were not written with interoperability in mind.

- Solutions contain code of unknown origin.

  After JKHL Enterprises deployed their first SOA solution to production, the Intellectual Property department contacted the project manager to ask about the origin of all the code in the deployed solution. It is extremely important, from a legal standpoint, to know the origin of the code and understand the licensing implications. The project manager contacted the development lead who was unable to fully determine the origin of the code. This manager asked developers, but a few developers at the time were contractors and could not be reached.

The WS-I compliance and certificate of originality policies respectively address these two issues. The scenario involves WebSphere Service Registry and Repository and Rational Asset Manager (Figure 6).

![Figure 6: Service lifecycle policy and WS-I compliance and certificate of originality policies](image-url)
The scenario follows this flow:

1. The SOA architect configures WebSphere Service Registry and Repository with a service life cycle that governs the Verify Credit service (and possibly other services) from inception to retirement. The architect defines a service life cycle where services are (in order): Outlined, Owned, Funded, Designed, Assembled, Tested, Deployed, Available, Retired, and Archived. This is described in Figure 7.

![Figure 7 Service life cycle for the Verify Credit service](image)

In this scenario, we look in more detail at the Funded to Designed and Designed to Assembled transitions.

The WS-I organization defines interoperability best practices, which are codified in WS-I profiles. (WS-I profiles extend Web services standards such as WSDL.) The WS-I Basic Profile addresses the interoperability issues encountered by JKHL Enterprises. For more information about the WS-I organization, refer to the following address:

http://ws-i.org/

2. In WebSphere Service Registry and Repository, the SOA architect authors a service lifecycle policy that requires compliance with the WS-I Basic Profile for Web services definitions. To do so, the architect selects the Funded to Designed service lifecycle state transition. The architect then selects the policies to associate with the transition. The WSDL service definitions must be WS-I compliant. The SOA architect then sees the policies associated with the Verify Eligibility service and the details about these policies.
The SOA architect configures WebSphere Service Registry and Repository to verify which policies are already enforced.

A certificate of originality certifies that the contents produced are the result of an entity’s own work and not materials taken from previously published assets. JKHL Enterprises decides to adopt a certificate of originality policy, which stipulates that, before the code can be deployed, developers must review their code and certify its origin.

In WebSphere Service Registry and Repository, the SOA architect repeats the previous step to associate the certificate of originality policy with the Designed to Assemble service lifecycle state transition.

3. In Rational Asset Manager, the development lead associates the WS-I Compliance policy with the assets in the JKHL Enterprises’ developer community, and specifies that the policy will be enforced when assets are published to the repository.

The development lead repeats the previous step to associate the certificate of originality policy with assets in the same community and likewise to ensure that the policy is enforced when assets are published.

4. In Rational Asset Manager, the development lead publishes a reusable software asset, which contains the Verify Credit WSDL definition, triggering the WS-I compliance policy enforcement by Rational Asset Manager. The WSDL is not WS-I compliant. As a result, Rational Asset Manager prevents the definitions from being published to WebSphere Service Registry and Repository.

In Rational Asset Manager, the development lead sees that an error occurred and sees the reason for the error (not WS-I compliant). The development lead modifies the WSDL to make it WS-I compliant. When it is compliant, the reusable software asset is approved.

5. In WebSphere Service Registry and Repository, the SOA architect transitions the Verify Credit service from Funded to Designed, triggering WebSphere Service Registry and Repository to verify that the WS-I compliance and certificate of originality policies have been enforced. The SOA architect can see that the policies have been enforced by Rational Asset Manager. The Verify Credit service transitions to the Designed state.

**Note:** It is also possible to register a service definition directly in WebSphere Service Registry and Repository, without previously registering in Rational Asset Manager. In that case, WebSphere Service Registry and Repository does the policy enforcement itself, instead verifying that it has been enforced.

In this scenario, we have seen how WebSphere Service Registry and Repository can be used to codify a service lifecycle (with states and transition). We have also seen how WebSphere Service Registry and Repository can be used to codify service lifecycle policies (policies associated with state transitions) related to build-time aspects. In addition, we have seen how Rational Asset Manager can be used to associate build-time policies to reusable software assets and enforce these policies. Finally, we have seen how WebSphere Service Registry and Repository can verify that a given policy had been enforced by Rational Asset Manager.

We illustrate the service lifecycle policy support with these two build-time policies (WS-I and certificate of originality). Although we do not describe it in detail in this paper, it is possible to have service lifecycle policies with runtime (operational) policies. For example, the following service lifecycle policy is supported:

“Before the Deployed to Available transition can happen, all messages must be secured.”

As you can see, the state transition happens at run time and refers to policies at the operational level (message security).
Message security policy

In this scenario, we address operational security concerns. The Verify Credit service that JKHL Enterprises originally implemented was designed to be used by outside consumers. These consumers included PartnerCo and other agencies, but JKHL Enterprises could not tell exactly who was consuming the service. Also, they could not tell if something happened to messages (that is, unauthorized modification) on their way to JKHL Enterprises. Specifically, they were faced with the following issues:

- Messages were sent in the clear, allowing application data to be viewed or changed by unauthorized parties.
- Messages were received from unknown message originators.
- Integrity or confidentiality of messages could not be checked.

To address these issues, the security architect was directed to design a message security policy that prevents application data from being viewed (integrity, confidentiality policy) by any unauthorized entity (message origin policy). The message security policy is for the Verify Credit service and is described as follows:

- The user name is included in messages.
- Messages are signed, and the header provides a hash that is checked for integrity by the receiver.
- Message bodies are encrypted.

The security architect has the responsibility to ensure that the messages sent to and from JKHL Enterprises meet these criteria. To do this, the security architect makes sure that the security policy is part of the service life cycle and that the appropriate enforcement points use the new policy.

This scenario involves Tivoli Security Policy Manager, WebSphere DataPower SOA Appliance, and WebSphere Service Registry and Repository (Figure 8).

![Figure 8](image-url)
This scenario follows this flow:

1. In Tivoli Security Policy Manager, the security architect imports the Verify Credit service from WebSphere Service Registry and Repository. The Verify Credit service has no associated policies.

   The security architect uses Tivoli Security Policy Manager to author a message security policy to be attached to the Verify Credit service. The security architect uses the Tivoli Security Policy Manager policy authoring tool to select a set of WS-SecurityPolicy assertions to include in a message security policy. These assertions include signature, encryption, and username token.

   In Tivoli Security Policy Manager, the security architect now associates the message security policy with the Verify Credit service.

2. The policy information is shared with WebSphere Service Registry and Repository. In WebSphere Service Registry and Repository, the SOA architect can see the policy definition and the association of that policy with the Verify Credit service.

3. WebSphere DataPower has been configured to be the message security policy enforcement point. WebSphere Service Registry and Repository uses the service and policy definition from WebSphere Service Registry and Repository to determine what must be enforced.

DataPower enforces the message security policy at run time.

In this scenario, we have seen that Tivoli Security Policy Manager can be used as a security policy authoring tool, where the authoring is based on assertion type selections. We have also seen how Tivoli Security Policy Manager can associate a security policy with a service. We have seen how the policy and association information can be exchanged with WebSphere Service Registry and Repository. In addition, we have seen how DataPower gets the service and policy information from WebSphere Service Registry and Repository and finally how DataPower enforces the message security policy at run time.

**Access control policy**

In this scenario, we address additional operational security issues. Because the Verify Credit service is reused by many inside and outside JKHL Enterprises, we must consider additional security aspects regarding the granularity of access. We describe security concerns in “Message security policy” on page 12. In this section, we focus on those security aspects that are related to unauthorized access.

As part of the security policy design effort, the security architect must address the following issues:

- Application data can be accessed by unauthorized parties.
- The originator of the service request is unknown.
- After consumers get in, they all have the same authorization.

The security architect has been instructed to develop a fine-grained authorization model with different roles having different access patterns. This model includes users and groups, resources, actions, and uses the existing infrastructure (for example, the user directories). In this scenario, you see how the components of the architecture support the authoring and enforcement of an access control policy that realizes this fine-grained authorization model.

At a high level, the access control policy can be described as follows. For the Get Credit service, anyone in the user role has access to their own information (credit score and so on).
For the Verify Credit service, only people in the agency role can access this service, and then they are unable to get the full credit information for people. They can only know whether a person is above a threshold when they submit their application.

In this scenario, the focal point is integration of fine-grained authorization with other security related information (users, resources, and so on). In the IBM portfolio, this type of security activity is realized with Tivoli Security Policy Manager, WebSphere DataPower, and WebSphere Service Registry and Repository (Figure 9).

This scenario follows this flow:

1. In Tivoli Security Policy Manager, the security architect has the capacity to define role-based authorization. The architect authors an access control policy and associate it with the Verify Credit service. The architect selects the appropriate role from the list of roles in Tivoli Security Policy Manager that were defined for JKHL Enterprises. The architect then authors the access control expression by using rules and conditions to permit or deny access to resources. This person uses the existing infrastructure in the configuration information for the policy. For example, the architect maps the user role in the policy expression to the user group in JKHL Enterprises’ LDAP. Attributes in the policy expression are mapped to their sources (for example in the message). After the policy is authored, the security architect attaches it to the Verify Credit service.

2. The security architect repeats the previous step for the Get Credit service's access control policy.

3. In WebSphere Service Registry and Repository, the SOA architect inspects the Verify Credit service definition and sees the associated policies, including the access control policy.

4. WebSphere DataPower is configured to be the enforcement point for the access control policy at run time. Moreover, DataPower is configured to call out to Tivoli Security Policy Manager to make a decision about the enforcement of the policy.

5. Users request the Verify Credit or Get Credit services. DataPower enforces the access control policy, asking Tivoli Security Policy Manager to make a decision based on the user privileges defined in the policy.
In this scenario, we have seen how Tivoli Security Policy Manager can be used to author an access control policy, by using the existing authorization infrastructure. We have also seen how the policy association can be viewed in WebSphere Service Registry and Repository. In addition, we have seen how DataPower can be used to enforce the access control policy, by using Tivoli Security Policy Manager as a policy decision point.

**Reliable messaging policy**

*Note: Because this scenario is similar in structure to the other scenarios, we provide an abbreviated description for it.*

JKHL Enterprises is faced with the issue of message loss due to a component failure. Reliable messaging addresses this problem by providing delivery assurance.

This scenario involves the SOA architect, WebSphere Service Registry and Repository, and DataPower (Figure 10).

![Figure 10] Reliable messaging policy scenario

This scenario follows this flow:

1. In WebSphere Service Registry and Repository, the SOA architect accesses the Verify Credit service definition and sees the list of policies that are currently associated with it. No reliable messaging policy is associated. The SOA architect selects the Verify Credit service endpoint and browses the WebSphere Service Registry and Repository library of policies. Policies are classified, and the architect is able to narrow down the list of applicable policies (reliable messaging), and select the exactly once reliable messaging policy. WebSphere Service Registry and Repository associates the policy with the service and produces an updated WS-Policy definition. The architect selects the Verify Credit service and now sees the associated reliable messaging policy.

2. The Verify Credit service is protected by the DataPower security gateway, and as such it has a proxy configured on the DataPower appliance. DataPower has been configured to poll WebSphere Service Registry and Repository for changes to the definitions of the services it proxies. DataPower notices that a change has been made to the Verify Credit
service definition in WebSphere Service Registry and Repository. DataPower automatically makes modification to the Verify Credit proxy configuration to include the updated policy (exactly once reliable messaging). DataPower then enforces the reliable messaging policy for any subsequent Verify Credit service invocations.

In this scenario, we have seen how WebSphere Service Registry and Repository can be used to associate a reliable messaging policy with a service. We have also seen how the policy information can be exchanged between WebSphere Service Registry and Repository and DataPower. In addition, we have seen how DataPower can be used as the reliable messaging policy enforcement point.

Conclusion

SOA policy management is an emerging area of interest with customers as they start to demonstrate some of the governance and service reuse principles and practices of SOA in their application run times. The IBM portfolio of products offers both a set of robust enforcement points over a wide range of policy domains and a set of extensibility points to progressively develop more advanced policy management capabilities over time.

Although showcased by using a simple implementation, the policy management capabilities discussed in this paper provide a solid foundation for service interoperability and governance. IBM intends to improve these functions with additional policy libraries and support the federation of policy across repositories and runtime enforcement points.
The Team that wrote this IBM Redpaper

Maryann Hondo is a Senior Technical Staff Member in the WebSphere Technical Institute at IBM working with WebSphere DataPower, SOAs, policy, and security. She previously worked in the IBM services organization with customers on SOA enablement focusing on security services. She is coauthor of the WS-Security, WS-Trust, WS-SecureConversation, WS-Policy and WS-Federation specifications produced by IBM and other business partners under the Web Services Security Roadmap. Maryann joined IBM Lotus® in 1996 where she managed the IBM Tivoli Jonah team (IETF PKIX reference implementation) and was the security architect for Lotus e-Suite participating in the development of Java™ Security (JAAS). Prior to joining IBM, she worked for Hewlett-Packard on DCE and PKI-based single sign-on, Digital on a B1/CMW operating system, and AT&T Bell Labs on the B2 UNIX® operating system.

Bertrand Portier is a Senior IT Architect for IBM Software Group, World-Wide SOA Technical Sales. He has worked for IBM for eight years in the development, services, and sales organizations, helping clients on such topics as Java, Web services, model-driven development, and SOA. He has presented and written extensively on these topics.

Franco Potepan is the SOA Governance Lead for the IBM Worldwide SOA Technical Sales team. In his current role, he defines and promotes business-driven development best practices for SOAs by using IBM tools and methodologies. He has 20 years of experience in software engineering, focusing on iterative process adoption, model-driven development, requirements management, object-oriented analysis and design, and architectural frameworks. Franco began his career in research, designing controls systems for synchrotrons at Elettra (Italy) and CERN (Switzerland). He then worked as a consultant, project manager and software architect. He joined the services organization of Rational Software, now IBM Rational, in 2001.

References

This section includes reference information for further reading materials:

- WebSphere Service Registry and Repository

- WebSphere DataPower SOA Appliances

- Rational Asset Manager

SOA Policy Management 17
- SOA Foundation Scenario Redpaper publications
- SOA Foundation Scenario IBM Redbooks® publications
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- DataPower®
- IBM®
- Lotus®
- Rational®
- Redbooks®
- Redbooks (logo)®
- Tivoli®
- WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.