Delivering SOA solutions: service lifecycle management.

Gary McBride, SOA go-to-market manager, Rational® software, IBM Software Group

Phil Fritz, SOA strategy, Tivoli® software, IBM Software Group

Brian Fialho, SOA marketing engineer, Rational software, IBM Software Group

Contents

Executive summary

With service-oriented architecture (SOA), once-siloed business functions, IT processes and data are exposed as services that can be reused—across departments, lines of business (LOBs) and even entire organizations. Many organizational groups, both internal and external, may help compose and execute business processes and frequently share certain processes in common. The unprecedented scope of an SOA initiative brings to the forefront a number of management and governance issues that were sidestepped in the past.

The key to a successful SOA implementation is managing and governing activities throughout the entire SOA delivery lifecycle by ensuring that services conform to the needs of all of the business's stakeholders. Ultimately, service lifecycle management allows the business to ensure that the process by which services are defined, created, tested, deployed, optimized and retired is manageable, repeatable and auditable. From development to operations, this process allows use of collaboration and automation technologies to deliver benefits such as better governance, lower costs, greater business flexibility and improved IT productivity.

Service lifecycle management is a service-oriented lifecycle management framework that establishes the service as the focal point of SOA governance, informing software development and delivery at every step. It helps ensure that the essential connections are made between teams responsible for defining, creating and delivering quality services. Diverse virtual teams can include business analysts, architects, project managers, developers, testers, release managers and operations personnel— many of whom may include

Highlights

partners, contractors and software application vendors. Service lifecycle management helps ensure that each service delivered matches the business need and, most importantly, that all services can be readily adapted, reused and evolved as needs change.

Service lifecycle management provides a flexible, open, structured way to improve SOA governance. Focusing on five key aspects—process and portfolio management, change and release management, quality management, architecture management and infrastructure management—addresses the comprehensive control and reporting functions a services delivery organization needs in order to extend policies and procedures to teams working throughout an expansive enterprise, including external partners and suppliers.

Service-oriented lifecycle management helps improve SOA governance by addressing comprehensive control and reporting functions.

With an established service lifecycle management process, organizations are better able to encourage faster adoption of service orientation across the enterprise, and more quickly realize a return on investment. A well-managed and better-governed SOA approach is essential. Service consumers from different departments will be more likely to use (and reuse) services and processes if their integrity, security, reliability and performance can be assured.

Understanding service lifecycle management

Effective governance helps ensure that projects align with business goals, meet quality expectations and get delivered on time.

Effective enterprises have interlocking governance structures at all levels. These structures define chains of responsibility, authority and communication that empower people within the organization. Measurement, policy and control mechanisms enable people to carry out their organizational roles and responsibilities. Within an IT organization, governance helps ensure that project investments align with business goals, and that the deliverables are

Highlights

SOA governance provides a solid framework for making critical decisions during service development projects.

monitored for quality and on-time delivery. Governance also enables managers to allocate and control project resources, clarify roles and responsibilities, and ensure a good return on IT investments. Effective IT governance also employs audit trails to better manage implementation and delivery risks, ensuring that the organization meets regulatory and compliance standards.

SOA governance

SOA governance has been identified as a critical component in the implementation of a successful SOA—and of crucial importance throughout all stages of the SOA lifecycle, during service modeling, assembly, deployment and management. Once an organization defines the business services required, it should consider questions such as:

- *What components do we already have in-house that we could reuse?*
- *What parts should we develop ourselves? What should we outsource?*
- *Can we buy any commercial off-the-shelf (COTS) applications from independent software vendors (ISVs)?*
- *How can we assemble all of the components into an effective service?*
- *How can we make sure the services will operate in the environments for which they are destined?*
- *How do we ensure the proper infrastructure and capacity for the service use cases we've developed? What are the contingency plans?*
- *How will we measure attainment of service-level objectives?*

SOA governance is the mechanism for ensuring that the organization has a solid structure for making these critical decisions. It allows the organization to better manage relationships between services and those who use them. At the same time, SOA governance can help ensure that all service-related activities comply with the laws, policies and standards the enterprise must follow—while enabling and maintaining audit readiness.

Highlights

A strong SOA governance structure facilitates collaboration among business groups and compatibility across processes and technologies.

Establishing a strong SOA governance structure may require some degree of organizational transformation to align roles and lines of authority with new policies, standards, procedures and processes. SOA demands collaboration among groups that may not have worked together previously and compatibility across formerly separate processes and technologies. Without these changes, an SOA implementation can quickly become siloed within each LOB, hindering the business from achieving the high degree of flexibility and responsiveness that an SOA is designed to deliver.

Service lifecycle management

Policies, standards and procedures are only as good as the care taken to implement them. Once an organization establishes an SOA governance structure, it's critical to get buy-in from everyone who must work within it. The goal is to help ensure that new directives are not too difficult to follow, preventing employees from reverting back to old practices that simply cannot support new technologies and approaches.

Through service lifecycle management, organizations can easily extend SOA governance structures to everyone involved in service creation and delivery.

Service-oriented lifecycle management plays a key role. It focuses on implementing the decision rights, processes and policies that govern service development, deployment and management, and then it focuses on automating those processes and monitoring the results. By using an established infrastructure that defines an end-to-end process for developing and managing services throughout the enterprise, organizations can easily extend their SOA governance structures to all of the people involved in service creation and delivery, regardless of an employee's office location, country or time zone. Business partners and suppliers with different organizational structures,

Highlights

processes and technologies can also be brought into the fold. And players from business analysts to testers can validate that the services they are modeling, assembling, deploying and managing remain aligned with enterprise needs.

In short, service lifecycle management can help you complete your SOA project quickly by establishing procedures and automated processes that ensure your SOA project is growing in the direction you expect it to. Success helps assure business units that it is safe to share their once-proprietary services and reuse enterprise assets, and even expand those assets to other business units or outside partners. By mining existing software and data investments to create new services, and by encouraging widespread use of these assets across the enterprise, organizations can quickly realize benefits, including greater productivity, process flexibility and organizational responsiveness, while lowering software and systems delivery costs.

IBM has identified five key aspects of SOA service lifecycle management, each of which crosscuts the entire SOA lifecycle.

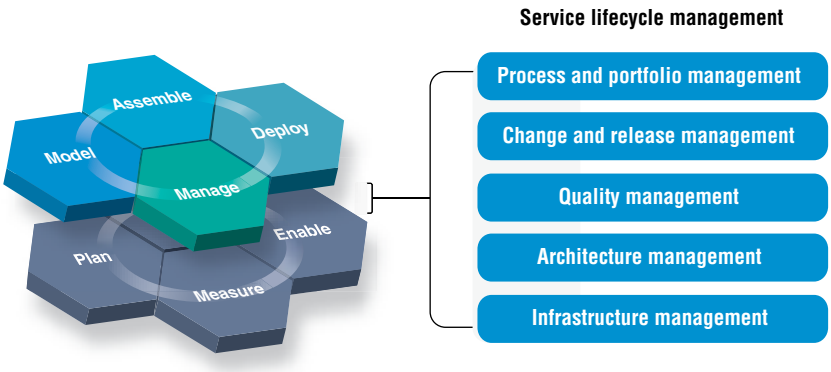

Figure 1: Five aspects of SOA service lifecycle management

Highlights

Spanning multiple technologies and environments, each aspect of SOA service lifecycle management highlights a crucial element of successful SOA.

Key considerations for service lifecycle management

IBM has identified five key aspects of SOA service lifecycle management, each of which crosscuts the entire SOA lifecycle. These aspects bridge multiple technologies and environments, and each highlights a crucial element of successful SOA.

- ***Process and portfolio management.*** *Analyze business needs to decide what shared services are needed and in which areas of the business. Prioritize and allocate resources for development and maintenance of shared services. Mine for available and would-be services. And help ensure alignment between services projects and business goals and delivery of expected results.*
- ***Architecture management.*** *Search for components and services that can be reused. Support standards and procedures to minimize risk when integrating new services into existing business processes. And support measures to protect against business interruption and privacy violations when mining existing mission-critical applications for delivery via SOA.*
- ***Change and release management.*** *Help ensure that all iterations and components of a service are interoperable and have complementary functionality; can be easily assembled and configured for the intended run-time environment; and remain aligned with business goals and requirements. Automate processes around issuing changes to applications. Track configurations of deployed services and composite applications, and automate processes associated with the release of new functionality.*

Highlights

- ***Quality management.*** *Support standards and procedures for validating the reliability, scalability and performance of single and composite services. Facilitate shared responsibility for maintaining best practices and quality measures. Provide mechanisms and reports to verify that a service has fulfilled all requirements before it is deployed. And assure that a service's behavioral characteristics remain stable and predictable throughout its lifecycle.*
- ***Infrastructure management.*** *Keep track of all SOA assets, including those that are planned, in development and in production. Monitor performance and other indicators about the IT environment, including resources, configuration items, user identities and interrelationships among entities. Help ensure infrastructure scalability and flexibility. Report performance against service-level commitments. Help secure access to services, and federate identities across a variety of application run times. Discover assets and track changes to configurations. Provide incident management and automate problem resolution. Provide dashboard views of IT services. And allow continuous optimization based on business drivers.*

Key aspects of service-oriented management occur at each stage of the service implementation process—from identification to connectivity.

Implementing service lifecycle management

This section explores the service-oriented management aspects within the context of each stage of the SOA lifecycle: model, assemble, deploy and manage.

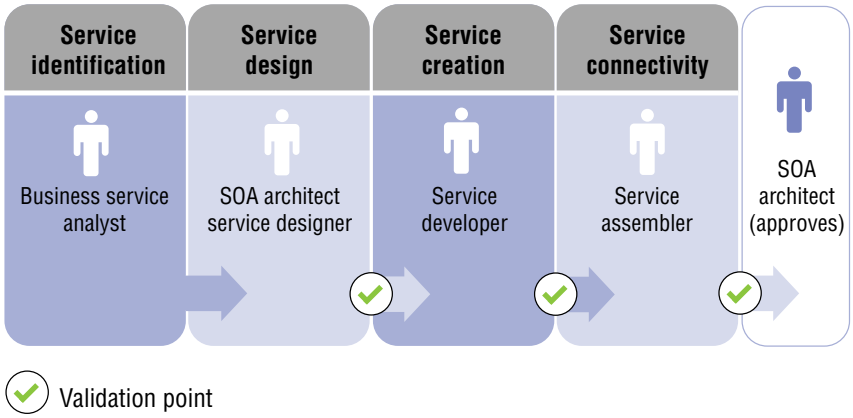

Figure 2: Service identification, design, creation and connectivity processes

Highlights

At the model stage, organizations assess, analyze and validate planned service development projects.

Model: from approval to design

Let's consider the way your organization would use SOA service lifecycle management. First, you would pursue a series of assessment and planning activities known as the model stage.

Annual planning for project and portfolio management

Annual portfolio planning exercises typically occur in the last quarter of the financial year. IT executives examine the lists of projects requested for the upcoming year to determine their validity, scope, cost and schedule estimates, among other things. Project requests are prioritized within each LOB according to a cost-benefit analysis for each project. Once that is completed, the projects are prioritized across the different LOBs.

As a result, IT executives work with their LOB counterparts to eliminate duplication and overlap between projects, and readjust the alignment of project goals with the company's strategic direction, business initiatives and priorities for the upcoming year. Ideally, the information captured in the early scope, cost and schedule estimates should be maintained so that when a project begins, key information is readily available to the project team. This also helps in measuring and monitoring planned figures versus actual performance to ensure that projects are not deviating significantly from estimates, thereby providing proactive indicators for troubled projects.

Modeling and analyzing business processes

To create a service, an important first step is to establish a business model to help teams analyze current processes. What process can we put in place so that new services are available in a timely way? How would that process relate to existing processes? Who should own this new process? What business unit should pay for it? Process models can be used to analyze a given business at a

Highlights

high level to validate transaction flows, and then exercises can be performed to optimize the processes. Once the processes have been validated and optimized, they can be used to identify and detail more granular use cases.

Early architecture management considerations

Before a new service can be implemented, functionality redundancies must be identified within the organization's existing architecture.

This is the stage to evaluate the proposed new service against a preliminary set of architecture management criteria. Existing IT assets and components need to be scanned to determine whether the desired functionality already exists or whether a new component needs to be implemented. In some cases where there are redundant systems, a decision needs to be made as to which system is the system of record. This will determine the final location of the service and its underlying implementation, which has implications with respect to service ownership, service-level agreements (SLAs), maintenance, and current and ongoing funding. These decisions may also be governed by policies documented in process guidance and asset/service repositories.

During the service identification process, business service analysts follow a series of steps, beginning with requirements gathering.

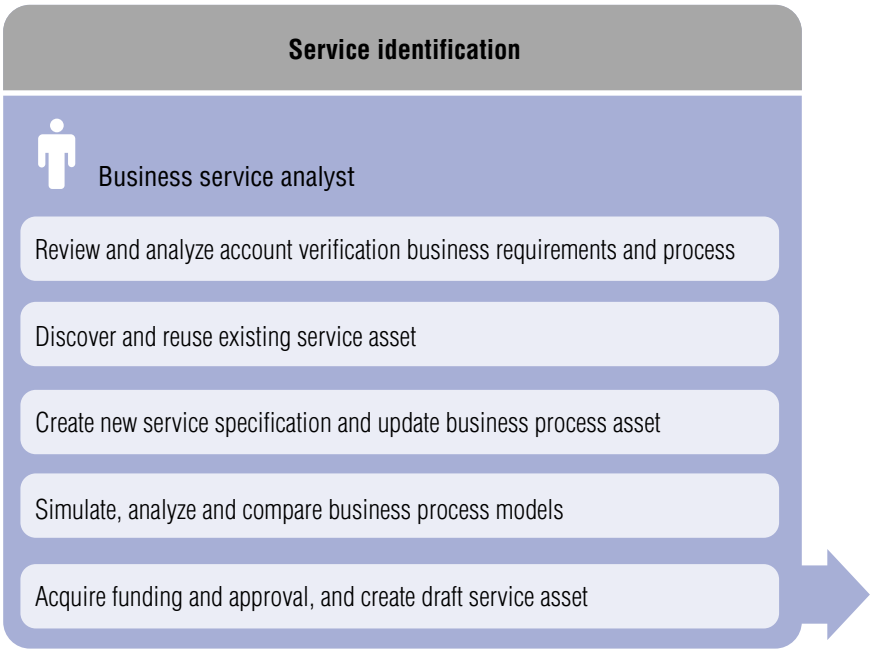

Figure 3: Service identification process

Highlights

For an SOA-bound service, business goals should align with business processes and trace to detailed requirements to help ensure alignment between business and IT.

Eliciting and managing requirements

For a traditional IT application, the creation process typically begins with making a list of the requirements, seeing what exists that can be reused, adding new functions, writing "glue" code to hold it all together and consulting with experts to make sure that the code complies with privacy, legal and other constraints documented in company business processes. However, for an SOA-bound service, you must focus on architecture management from the very beginning, when you're gathering and specifying requirements. To design the service correctly, you must consider the business processes likely to use the service. Of course, you may not anticipate all possible future uses, but you can make reasonable guesses by looking at your business process models as well as other documents that influenced the decision to develop the service. Put another way, requirements and anticipation are factors of not just business models, but also existing IT assets.

Ideally, high-level business goals should align with business processes and trace to more detailed requirements, helping to ensure alignment between business and IT. Taking this one step further, it is also very important to be able to link from requirements to key design elements in the Unified Modeling Language (UML) architecture and design diagrams. This way, as requirements change, you can be sure that nothing will slip between the cracks since dependents affected by the change can be highlighted.

Modeling the service

A use case is a sophisticated type of requirement that can help direct service creation efforts. It tells the story of how the service will interact with users, encompassing a series of scenarios that vary according to user behavior. Because use cases focus on user actions, they provide the basis for interface design: screen appearance, content and navigation. They also enable developers to specify objects and classes, attributes and operations from a high-level

Highlights

To create a new service, SOA architects and service designers often use cases to inform interface design and specify attributes and operations.

perspective that everyone on the team can understand. These specifications—called use-case realizations—depict how objects interact to implement each possible scenario. Developers can also create traceability between requirements and design elements from within these modeling tools.

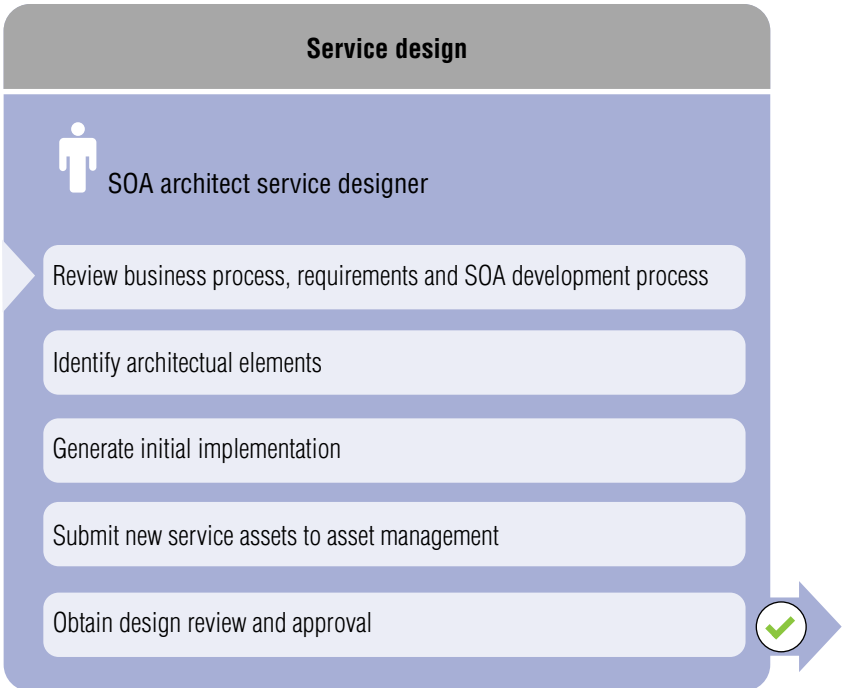

Figure 4: Service design process

Searching for reusable assets and allocating resources

At this stage, you search for reusable assets that you might use to assemble all or part of the new service. This step includes identifying metadata based on the Reusable Asset Specification, making it fast and easy to find candidate assets that will interoperate with other services you develop and run on your target platform. Once you have a well-specified, requirements-based model,

Highlights

the project manager develops a project plan that includes a breakdown of work, resources and scheduling information. Later, as the development effort proceeds, the project manager can monitor progress and share the information with all stakeholders, viewing actual costs and other indicators, and then making necessary adjustments.

At the assemble phase, developers bring together reusable service assets to create service-oriented applications that automate and integrate business processes.

Assemble: from design to test

At this step, developers assemble the reusable service assets to create service-oriented applications that automate and integrate business processes.

Refining the service architecture

Next, you conduct a series of service lifecycle activities encompassing architecture design, service and process development, and testing. Once you have completed the model activities in the lifecycle, you refine the service model and flesh out the architecture with actual services, frameworks and business processes.

During the creation process, developers conduct service lifecycle activities encompassing architecture design, service and process development and testing.

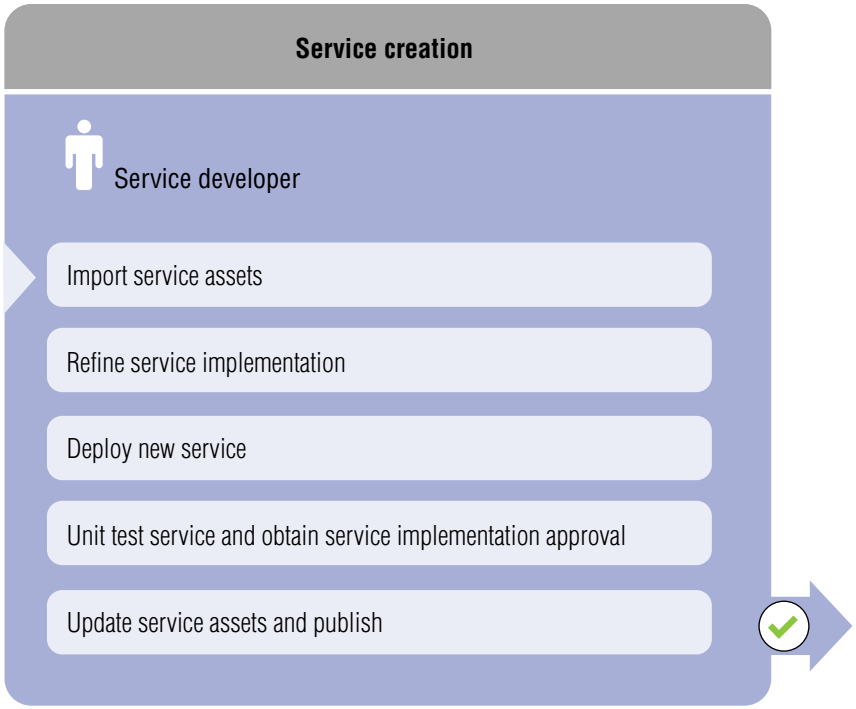

Figure 5: Service creation process

Highlights

Creating the service

By now you will have determined which services will be reused, modified and created from scratch. Based on your SOA governance structure and process, as well as issue-tracking assignments, you can grant selective check-in and checkout privileges to different team members as they create code, assemble components and correct defects.

At this stage, you may opt to integrate automated build and release management software that enables you to reduce errors and speed readiness as you deploy single and composite services.

To help ensure that a new service will work correctly, developers test services separately and again after assembling the composite application.

Testing for use and reuse

In a service-oriented environment, your goal is to create assets that satisfy more than one need and that can be applied to a number of business processes. As for a traditional application, you must first test each component separately, and then test again after assembling the composite application. This process helps ensure that the service will operate reliably and efficiently within the environments for which it was designed. The best way to achieve this is to write test cases directly from use cases. Testers create test cases early in the project, before any code is written, and they begin testing the first prototypes to verify alignment with requirements. This way, testers gain a clear understanding of the service and a logical basis on which to run a well-governed testing process. The test cases must also address the nonfunctional characteristics of a service, for example, its response times, capacity, dependencies and security configuration. As the service gets reused, more importance will be placed on these nonfunctional aspects (e. g., Can it handle the new load or usage pattern? Will it support different authentication mechanisms?).

Highlights

Through guerilla testing, testers can support successful service reuse by testing service performance beyond anticipated business conditions.

When it comes to services, developers and testers must step outside traditional boundaries and perform what practitioners refer to as *guerilla testing*. Basically, this is testing with reuse in mind. It involves compiling a set of tests for conditions that you do not expect the service to be exposed to in the normal course of business, keeping in mind that you cannot anticipate future uses. It can be more cost-effective to apply a little imagination and prepare the service for such unanticipated conditions now rather than having to rework them later—particularly if the original service creation team is no longer available. On the plus side, guerilla testing (and guerilla design) can be fun. It allows developers and testers to think creatively about future technology possibilities. However, standards, criteria and best practices are also necessary to keep the testing aligned with schedule, budget and business constraints.

The testing phase also can help you decide when and how to instrument the service (or the environment that hosts the service) for management. In fact, many of the same characteristics associated with testing the service can be used for operational monitoring—making the testing stage an ideal time to define service performance baselines, identify and classify exceptions, and define events and/or logs to be used for future debugging and problem determination. Given the importance of nonfunctional characteristics with increasing reuse, it is critical to test these characteristics and leverage the test instrumentation during the operational phase of the lifecycle.

Verifying process compliance

Service lifecycle management processes provide progress metrics that enable project managers to enforce process compliance at every step.

With ongoing visibility into project progress, managers can help ensure process compliance at every step. To this end, the integrated service lifecycle management processes can provide a constant flow of metrics and reports for project checkpoints and other monitoring and control mechanisms. Traceability to requirements throughout the service creation process ensures that the service will be compatible with other business processes, that it will comply with external regulations, that it will meet enterprise SOA architectural requirements

Highlights

and that it will run on multiple technology platforms. By tracing requirements, you can also provide an effective way to judge whether a new service is ready for deployment: Once you verify that it satisfies all requirements, the service resides in a central repository, waiting for calls to fulfill its role in the business process.

Deploy: from test to the SOA service environment

Once a service is declared "fit for service," a whole new set of governance challenges comes into play. What are the rules for promoting the service to other departments—and to parties external to the enterprise? Who will manage the system(s) on which the service will run? Who is responsible for managing service relationships? How will problem determination be performed? How can you analyze the impact of the new service on existing processes—and on the bottom line? How will service changes be dealt with? How are those changes released in a controlled fashion? When changes occur, how are they discovered or reconciled with the desired state?

During service deployment, organizations must support a critical set of tasks, roles and functionality.

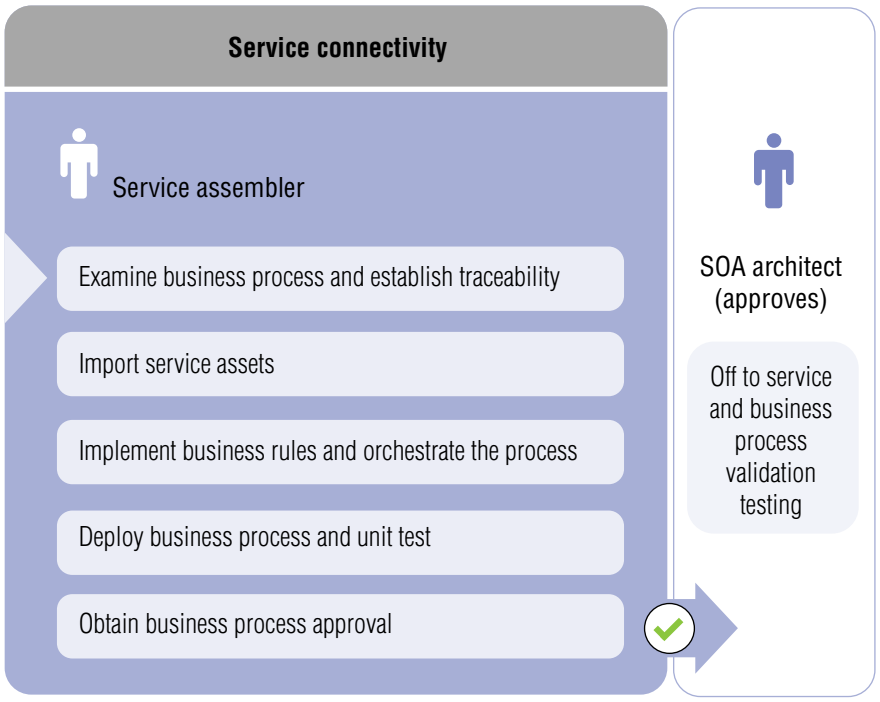

Figure 6: Service connectivity process

Highlights

IT staff can use metadata to evaluate whether service deployment was successful and observe service usage patterns.

As the focus shifts from development to deployment, you must consider a different set of tasks, roles and functionality. As developers and asset owners publish the relevant service metadata, IT operations can leverage that metadata to evaluate it against the monitored information from their production systems. This shared metadata can be used to evaluate whether deployment of the service was carried out successfully and to ascertain the service usage patterns. Plus, service information contained in registries can be reconciled with the actual monitoring data collected from production systems—establishing whether there are services in deployment that have not been registered or that have not gone through established management processes.

Manage: from operation to retirement

Once services are in production, organizations must manage them to help support service quality, security, compliance and adherence to service-level objectives.

Once in production, services need to be monitored, secured and managed to ensure that the organization delivers proper service quality, that the service adheres to security policies, that it meets service-level objectives and that it complies with regulatory and corporate mandates. Change management continues to be of vital importance, as SOA enables greater flexibility in allowing services to be reused (as well as masking changes within the implementation environment or other infrastructure). This higher degree of flexibility, while providing benefits to the business, implies much higher rates of change in IT operations—and increased demand for tracking dependencies and tracking changes over time. For IT departments, areas of concern for SOA deployments include:

- *Automating the discovery and dependency tracking of services to the underlying IT infrastructure so that the IT team can handle more services with the same headcount.*
- *Monitoring and managing at the service layer of abstraction to provide linkage to IT resources and insight into the business processes the services support.*

Highlights

- *Publishing performance and other information to repositories to aid in service selection during run time and to establish the IT infrastructure dependencies on which services rely.*
- *Securing services and providing proper identity federation.*
- *Aggregating events and monitoring information to provide a business-level view—in terms of business services or SLA reporting—that can instantly provide an end-to-end view of the SOA-based composite application and allow for appropriate business impact.*
- *Automating the change and release management processes, ensuring that appropriate handoffs occur between development and IT operations while optimizing the workload of the staff now entrusted with service lifecycle management.*
- *Making operational data available to application support teams and developers for problem isolation and resolution. One of the key differences in the highly distributed applications enabled by SOA is the fact that, in many cases, problems cannot be replicated in nonproduction environments. Therefore, developers, application support staff and testers may need access to production monitoring information so that they can assess, isolate and fix problems throughout the production system.*

To avoid downtime and meet service-level commitments, organizations must monitor run-time performance throughout the service lifecycle.

A run-time service must be closely monitored and managed throughout its lifecycle—from its glory days of fulfilling calls for multiple uses, to the day it retires to the central repository to be reused. Consistent performance recording and reporting are necessary, both to detect bugs that may have slipped through the testing process and to keep up run-time service-level commitments. Especially for popular services that enjoy increasing usage over time, organizations must enforce governance policies and make sure that runs are carefully quantified and invoiced, if applicable.

SOA service lifecycle management can help organizations manage the governance policies, procedures, rules and standards required to help an SOA deliver business benefits.

Conclusion

Introducing SOA into an IT organization demands both structural and policy changes, as it brings new demands for alignment among multiple processes within different areas of the business. The governance policies, procedures, rules and standards required to help an SOA deliver business benefits—greater flexibility, productivity and financial returns—are complex and can quickly become overwhelming without adequate infrastructure support. SOA service lifecycle management is the answer, providing tailored, end-to-end support for the diverse teams that create, deploy and evolve services. IBM offers a complete array of products and services for service lifecycle management that automate activities within the discipline. Collectively, they represent the industry's most far-reaching and comprehensive infrastructure for implementing a successful SOA.

For more information

To find out more about SOA governance and service lifecycle management, please visit:

ibm.com/soa

or

ibm.com/soa/gov

or

ibm.com/rational/soa

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
12-07

RAW14006-USEN-00