SERVICE TECHNOLOGY SERIES FROM THOMAS ERL

The Top-Selling, De Facto Guide to SOA

Now Updated with New Content and Coverage of Microservices!

SECOND EDITION

Service-Oriented Architecture

Analysis and Design for Services and Microservices

ServiceTech PRESS

Thomas Erl

With contributions by Paulo Merson and Roger Stoffers

Service-Oriented Architecture

The following are examples of common Service Testing considerations:

- What types of service consumers could potentially access a service?
- Will the service need to be deployed in a cloud environment?
- What types of exception conditions and security threats could a service be potentially subjected to?
- Are there any security considerations specific to public clouds that need to be taken into account?
- How well do service contract documents communicate the functional scope and capabilities of a service?
- Are there SLA guarantees that need to be tested and verified?
- How easily can the service be composed and recomposed?
- Can the service be moved between on-premise and cloud environments?
- How easily can the service be discovered?
- Is compliance with any industry standards or profiles (such as WS-I profiles) required?
- If cloud deployed, are there proprietary characteristics being imposed by the cloud provider that are not compatible with on-premise service characteristics?
- How effective are the validation rules within the service contract and within the service logic?
- Have all possible service activities and service compositions been mapped out?
- For service compositions that span on-premise and cloud environments, is the performance and behavior consistent and reliable?

Because services are positioned as IT assets with runtime usage requirements comparable to commercial software products, similar quality assurance processes are generally required.

Service Deployment and Maintenance

Service deployment represents the actual implementation of a service into the production environment. This stage can involve numerous interdependent parts of the underlying service architecture and supporting infrastructure, such as:

- Distributed components
- Service contract documents
- Middleware (such as ESB and orchestration platforms)
- Cloud service implementation considerations
- Cloud-based IT resources encompassed by an on-premise or cloud-based service
- Custom service agents and intermediaries
- System agents and processors
- Cloud-based service agents, such as automated scaling listeners and pay-for-use monitors
- On-demand and dynamic scaling and billing configurations
- Proprietary runtime platform extensions
- Administration and monitoring products

Service maintenance refers to upgrades or changes that need to be made to the deployment environment, either as part of the initial implementation or subsequently. It does not pertain to changes that need to be made to the service contract or the service logic, nor does it relate to any changes that need to be made as part of the environment that would constitute a new version of the service.

Service Usage and Monitoring

A service that has been deployed and is actively in use as part of one or more service compositions (or has been made available for usage by service consumers in general) is considered to be in this stage. The ongoing monitoring of the active service generates metrics that are necessary to measure service usage for evolutionary maintenance (such as scalability, reliability, etc.), as well as for business assessment reasons (such as when calculating cost of ownership and ROI).

Special considerations regarding this stage apply to cloud-based services, such as:

- The cloud service may be hosted by virtualized IT resources that are further hosted by physical IT resources shared by multiple cloud consumer organizations.
- The cloud service usage may be monitored not only for performance, but also for billing purposes when its implementation is based on a per-usage fee license.
- The elasticity of the cloud service may be configured to allow for limited or unlimited scalability, thereby increasing the range of behavior (and changing its usage thresholds) when compared to an on-premise implementation.

This phase is often not documented separately, as it is not directly related to service delivery or projects responsible for delivering or altering services. It is noted in this book because while active and in use, a service can be subject to various governance considerations.

Service Discovery

To ensure that reusable services are consistently reused, project teams carry out a separate and explicitly defined service discovery process. The primary goal of this process is to identify one or more existing agnostic services (such as utility or entity services) within a given service inventory that can fulfill generic requirements for whatever business process the project team is tasked with automating.

The primary mechanism involved in performing service discovery is a service registry that contains relevant metadata about available and upcoming services, as well as pointers to the corresponding service contract documents (which can include SLAs). The communications quality of the metadata and service contract documents play a significant role in how successfully this process can be carried out. This is why the Service Discoverability (300) principle is dedicated solely to ensuring that information published about services is highly interpretable and discoverable.

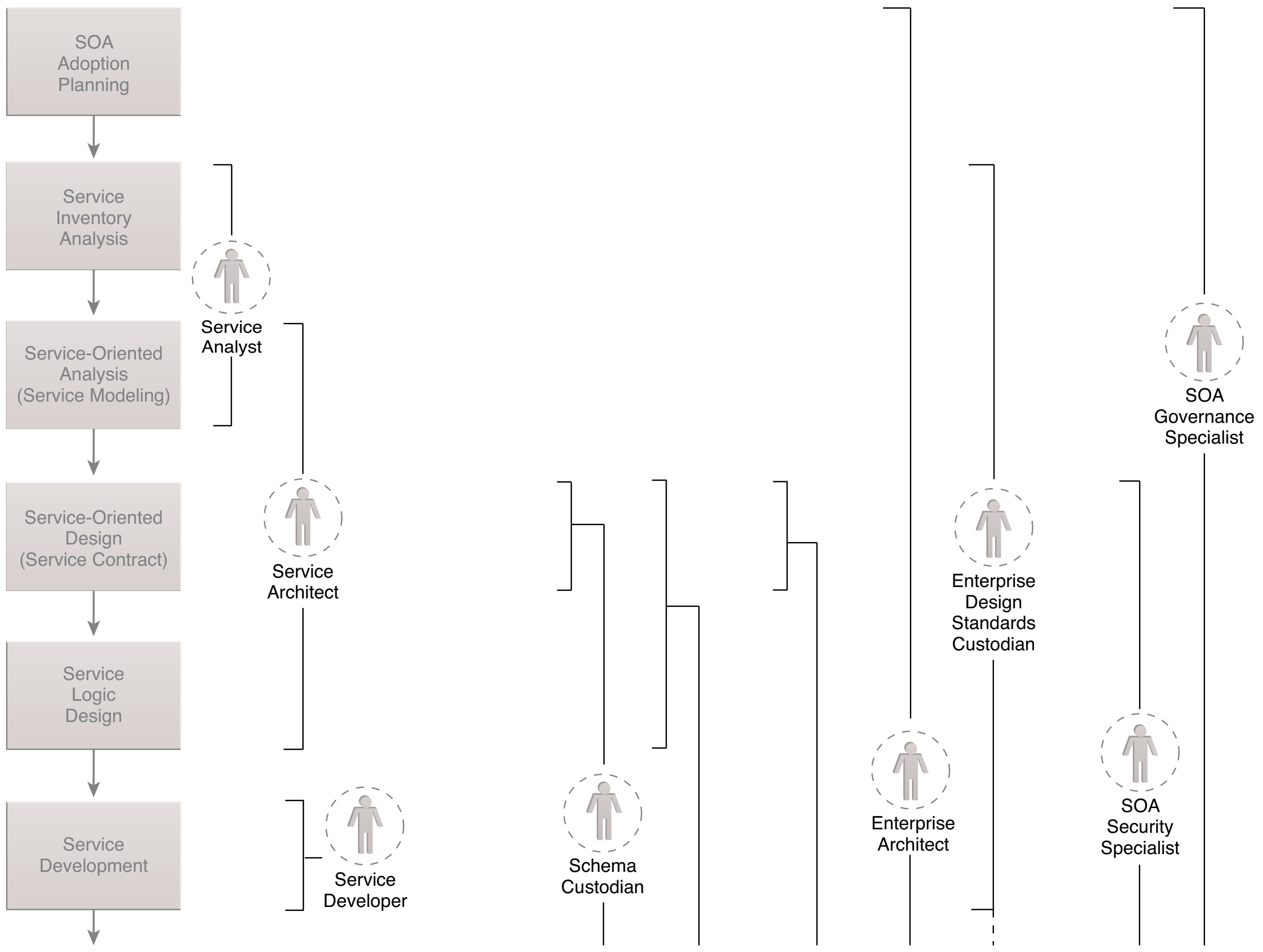
Service Versioning and Retirement

After a service has been implemented and used in production environments, the need may arise to make changes to the existing service logic or to increase the functional scope of the service. In cases like this, a new version of the service logic and/or the service contract will likely need to be introduced. To ensure that the versioning of a service can be carried out with minimal impact and disruption to service consumers that have already formed dependencies on the service, a formal service versioning process needs to be in place.

There are different versioning strategies, each of which introduces its own set of rules and priorities when it comes to managing the backward and forward compatibilities of services. (Chapter 10 provides fundamental coverage of common service versioning approaches for Web services and REST services.)

Project Stages and Organizational Roles

Figure 4.31 revisits the SOA project stages and maps them to common organizational roles. These roles are described in the *SOA Governance: Governing Shared Services On-Premise & in the Cloud* text book.


SOA Adoption Planning
Service Inventory Analysis
Service-Oriented Analysis (Service Modeling)
Service-Oriented Design (Service Contract)
Service Logic Design
Service Development
Service Analyst
Service Architect
Service Developer
Schema Custodian
Enterprise Architect
Enterprise Design Standards Custodian
SOA Security Specialist
SOA Governance Specialist

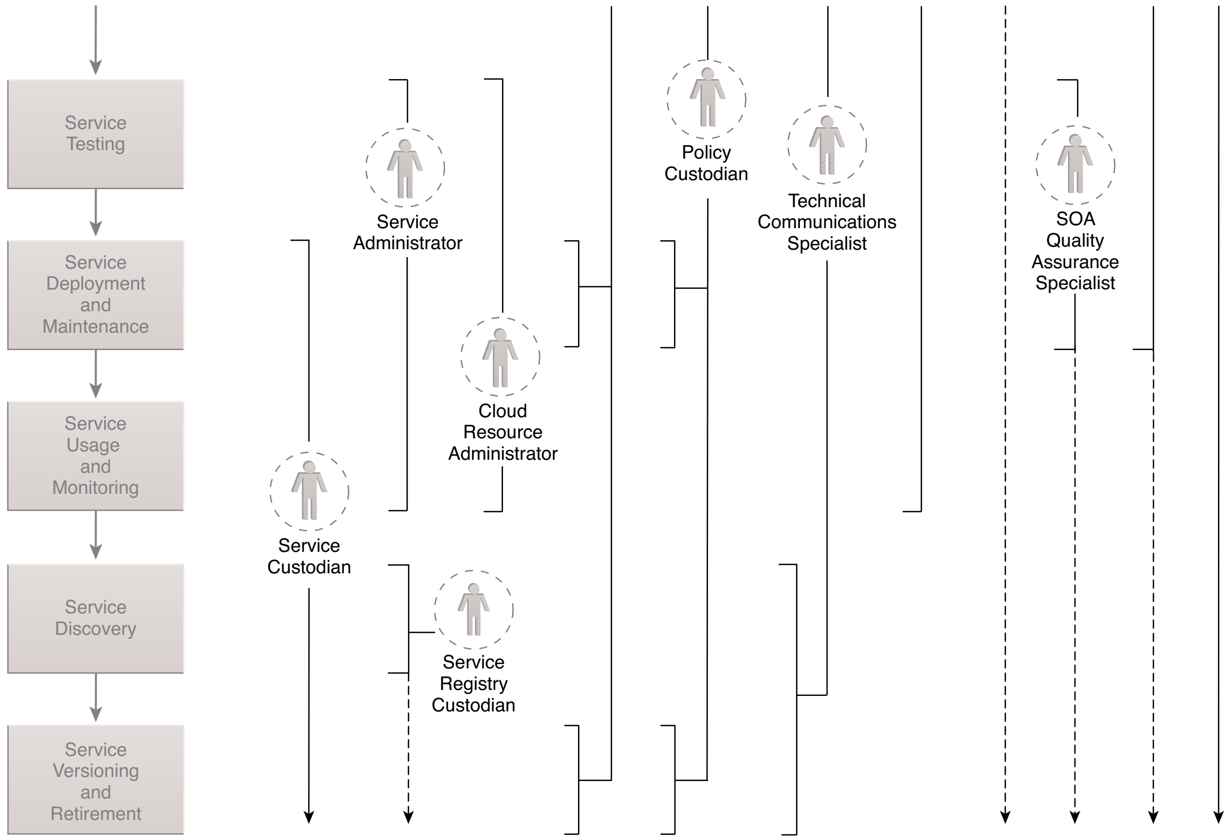

Service Testing
Service Deployment and Maintenance
Service Usage and Monitoring
Service Discovery
Service Versioning and Retirement
Service Administrator
Cloud Resource Administrator
Service Custodian
Service Registry Custodian
Policy Custodian
Technical Communications Specialist
SOA Quality Assurance Specialist

Figure 4.31

Shown here are common associations of organizational roles with different SOA project stages.