
Abstract

SOA is relatively new, so companies seeking to implement it cannot tap into a wealth of practical expertise. Without a common language and industry vocabulary based on shared experience, SOA may end up adding more custom logic and increased complexity to IT infrastructure, instead of delivering on its promise of intra and inter-enterprise services reuse and process interoperability. To help develop a shared language and collective body of knowledge about SOA, a group of SOA practitioners created this SOA Practitioners' Guide series of documents. In it, these SOA experts describe and document best practices and key learnings relating to SOA, to help other companies address the challenges of SOA. The SOA Practitioners' Guide is envisioned as a multi-part collection of publications that can act as a standard reference encyclopedia for all SOA stakeholders.

1.1 Intended Audience

This document is intended for the following audience:

- Business and IT leaders, who need to start and manage an SOA strategy across the enterprise/LOB
- Enterprise Architects who need to drive the vision and roadmap of the SOA program and the architecture of each implementation that falls under it
- Program Managers who need to manage a portfolio of sub-projects within an overall SOA business strategy
- Project Team Members, who need to map dependencies and develop a timeline that meets the business expectations
- Vendors who provide solutions and tools for new business capabilities to the business and IT
- Standards bodies which need a better understanding of use cases of how business and IT plan to leverage technology to meet their objectives.

Services Lifecycle Stages

1.2 Service Development

1.2.1 Actors

- Project manager
- Architects
- Development teams
- Release management
- IT operations

1.2.2 Tools Used

- IDE (for example, Visual Studio, Eclipse, JBuilder, JDeveloper, and BEA Workshop)
- Packaged application development tools
- Regression and performance testing tools
- Build tools (for example, Maven, Cruise Control, ANT, and shell scripts)
- Source control management systems

1.2.3 Artifacts (Deliverables)

- Source code
- Product specific metadata for configuration as well as service execution
- Java documents
- Release notes

1.2.3.1 Artifact Description

The artifacts produced during this stage include service definition, configuration, contract, and application configuration, as well as management, security and business policies at the service, application, department, and enterprise level. The application configuration also includes implementation configuration.

1.2.4 Service Lifecycle Stage Key Considerations

Tasks for this stage of service lifecycle include:

- Define criteria for when to outsource development of a service
- Perform technology review process prior to using a new technology within IT
- Assess skills required for the entire lifecycle of the service prior to development of service
- Map each service back to the business requirement
- Evaluate existing services to eliminate duplicate development.

1.2.5 Service Lifecycle stage recommended process

Developers can start developing services as soon as project managers or architects provide them with the design documents. These could be office documents or models based on enterprise architecture standards.

Developers can also start developing or modifying services as soon as business or IT operations opens a change request(CR). In this case the architect may not be involved. The application support technical lead or developer will review the change request and make the changes. Depending on the level of change and the IT PMO process, governance may be triggered when the developer uploads the metadata change to the SOA repository.

Service creation typically follows this process:

1. The application design or the change request is approved for development.
2. In case of new development, the designer may already have created service templates and assigned them to the developer. The developer downloads the service templates and completes them.
3. In case of new development where no templates exist, the developer either uses a development tool to create the service or searches for a similar service in the SOA repository and copies it over as a template.
4. In case of a change request, the developer downloads the code and modifies it.
5. In all cases, the developer also downloads the requirements and design artifacts for review and may modify them after clarifications from architects, project managers, or business users. The service governance process is triggered whenever a developer modifies the requirements or the design artifacts.
6. If the architect has wired services based on SCA, the developer could leverage an SCA-based tool to double click on the component to confirm that the service is created.
7. If the service being created is to be consumed by services being developed by the other teams, the developer should define the service and develop and configure the service simulation. The service producer does this by defining responses to sample requests and publishing them for consumption. This enables consuming services to test out the contracts prior to the service actually being developed.
8. If the service being created is consuming a service being developed by the other teams, the developer can leverage the service simulations produced by the other team.
9. The developer goes through multiple interactions to develop and test the services. As the regression test is generally end-to-end, the developer can either use the simulated testing environment for unit-level testing or move the service through the various testing stages.
10. The developer periodically synchronizes the metadata with the SOA repository. The SOA repository validates all the metadata before accepting it and also triggers the service governance process, if required.
11. The developer generates documentation, such as Java documents and release notes, that are synchronized with the SOA repository.

This describes the traditional services creation and maintenance process. Most software vendors now also provide business users the ability to create composite applications using a portal. Business users can create composite applications by loosely coupling various user interactions and interactions processes, and dynamically creating forms based on service definitions. The business user can perform most of these tasks directly in production. Tools provide the capability to develop, test, and deploy the service, and synchronize the metadata that drives these composite applications to the SOA repository.

Contributing SOA Practitioners

Surekha Durvasula, Enterprise Architect, Kohls

Martin Guttman, Principal Architect, Customer Solutions Group, Intel Corp

Ashok Kumar, Manager, SOA Architecture, Avis/Budget

Jeffery Lamb, Enterprise Architect, Wells Fargo

Tom Mitchell, Lead Technical Architect, Wells Fargo Private Client Services

Burc Oral, Individual Contributor

Yogish Pai, Chief Architect AquaLogic Composer, BEA Systems, Inc.

Tom Sedlack, Enterprise Architecture & Engineering, SunTrust Banks, Inc.

Dr Harsh Sharma, Senior Information Architect, MetLife

Sankar Ram Sundaresan, Chief Architect e-Business, HP-IT