One of the advantages of using a Service-Oriented Architecture (SOA) is the alignment of the IT systems to the business they are serving. This has an effect on the people who develop and operate these IT systems both in terms of the tasks they perform and the knowledge and skills they require. This article walks you through a simple integration scenario to illustrate how a team creates and runs a services-oriented solution. It uses user roles to describe the skills and responsibilities of the people involved, and it's geared toward technical leaders to help you understand how to organize the work associated with service-oriented solution development.

### View more content in this series

### Introduction

Every organization is different. It has its own culture, structure, skill pool, and assets. Yet many are struggling with similar issues:

- How to improve customer service and adapt to their changing needs
- How to be cost-efficient
- How to collaborate with customers, partners, competitors, and suppliers

Solving these issues requires change, and as the organization changes, so must the IT systems that underpin it.

There has been much written about the value in using an SOA to manage change in IT systems. SOA requires each IT application to have well-defined interfaces to represent the business function that it supports. These interfaces, or services, provide a catalog of IT functions that can be called as and when needed. The calls can be directly between applications, via workflow applications, or dynamically routed through an Enterprise Service Bus (ESB). If it's done well, a service-oriented approach allows an organization to structure and connect its IT systems together in a flexible manner that maps closely to the needs of the organization.

This article explains how organizations adopt an SOA. Given the variability between organizations, it explains the process in terms of user roles. These are characterizations of typical positions in an organization. A user role does not necessarily equate to an individual. A single person or a team could perform the user role. Similarly, a person may perform more than one role.
With user roles, we can describe how responsibilities are distributed in the organization, where communication and decisions are required, and the types of skills involved. From this, an organization can plan how to deploy people to fill these roles depending on the workload specific to their organization.

This article walks through a simple scenario where an organization decides to use an SOA for a new IT solution. Through user roles, it describes how an organization creates and runs a service-oriented solution. We’ve broken it down according to these topics:

- **Simple integration scenario** describes the solution scenario.
- **Put the user roles to work** explains how the solution is built and managed.
- **Summary and conclusions** briefly sums up the service-oriented approach.

**Simple integration scenario**

**Figure 1** shows the scenario used in this article. Although very simple, it illustrates many of the characteristics of an SOA.

For historical reasons, the organization has half of their customer data stored on one IT system and the rest on another. These two systems operate independently. The plan is to create a single service for customer data. When an application calls this service, a service routes the request to the appropriate back-end system.

Over time, the location of customer data may change or be enriched from other systems, so the solution requires a flexible implementation. Each back-end system provides a service for extracting the data it owns. A mediation running in the ESB supports the routing from the common service to the service for the appropriate back end.

**Figure 1. Simple integration scenario**

The following section describes how the user roles work together to design, develop, and run this solution.

**Put the user roles to work**

Creating the integrated service interface involves the selection of the appropriate approach followed by developing and deploying the solution, and then running the solution. This is coordinated through service-oriented governance. These phases correlate to the SOA life cycles of Model, Assemble, Run, and Manage.
The following sections describe which roles are involved in each phase and what they do.

Select the approach

Requirements for new solutions don't appear from nowhere. The Enterprise Architect, Business Analyst, and System Analyst are the roles that identify, prioritize, and group these needs into projects.

Enterprise Architect
The Enterprise Architect is responsible for the technical strategy of the organization and is the keeper of the SOA vision, which advocates think strategically, act tactically. Rather than commissioning huge projects to rip and replace the existing systems, the Enterprise Architect favors the ongoing rollout of small enhancements as a means to migrate to the desired structure.

Business Analyst
The Business Analyst is searching for ways to improve the performance of the business. This may involve changes to the way people work together, the tools and procedures they use, or the IT systems that support them. In this scenario, the Business Analyst has seen the opportunity to provide a new offering to customers. This requires a change to the IT systems, so the Business Analyst works with the System Analyst to look at the feasibility of supporting this new offering.

System Analyst
The System Analyst converts business requirements into requirements on the IT systems.

After talking to the Business Analyst, the System Analyst realizes that this new offering requires a consistent means to access the organization's customer data. Currently this data resides in two systems, each with a different interface and data format. Creating a new customer data repository is an expensive option because of the impact on the existing systems. So after discussions with the Enterprise Architect, the System Analyst proposes a common service interface for the customer data. Behind this interface, requests are routed to the existing systems.

This approach gives them the flexibility to merge the two customer data systems in the future without altering the new customer service applications, because the Common Customer Data Service isolates calling applications from this implementation detail.

The System Analyst's specification of the IT solution for the new offering contains enough detail for the organization to make a business decision about whether to invest in the new IT solution. If the decision is favorable, a project team is formed to develop the solution.

Develop the solution

IT solutions require a mixture of hardware, software, applications and configuration. This article focuses on the software aspects of the solution.

Software Architect
The Software Architect is responsible for dividing the required function between the components that make up a software solution. This person works with the specifications and standards used by
existing IT systems and determines where enhancements or where new components need to be written.

The Software Architect takes guidance from the Enterprise Architect to ensure the architecture of a project matches the overall direction of the IT strategy. The requirements for the solution come from the System Analyst.

In this scenario, the presence of the Common Customer Data Service divides the new solution into two halves:

1. The application code that provides the new function to customers calls the Common Customer Data Service.
2. The logic routes the Common Customer Data Service requests to the appropriate back-end system.

The Software Architect chooses to use a Java 2 Platform, Enterprise Edition (J2EE) application to implement the new application code. Behind the Common Customer Data Service, the Software Architect defines a service interface for each of the back-end systems (to clearly define how these systems are being called) and specifies that an ESB mediation maps the requests and replies between the two levels of service interface.

**Developer**
The Developer designs and implements pieces of the solution and tends to have specialized skills that focus on a development platform, programming language, and/or business area. The following types of developers are typically involved in building service-oriented solutions.

- **Application Developer**: The Application Developer understands the business area for a solution and implements the application code that performs the business-related function -- in this case the new J2EE application that calls the common customer data service. Application Developers need to understand their application development environments (such as J2EE) and how to call a service from within those environments. They work from a specification of the service interface provided by either the Software Architect or another Developer.
- **Component Developer**: A Component Developer codes self-contained chunks of code called **components**. These components are designed to be reused in multiple applications. In this scenario, the Component Developer could code the mediation for the Common Customer Data Service.
- **Integration Developer**: An Integration Developer is responsible for building services by configuring components and linking them together. These components could have been written by a Component Developer or provided as part of an ESB product. Integration Developers typically have a good understanding of integration techniques and patterns but limited programming skills. If an integration scenario requires some complex programming logic, an Integration Developer works with a Component Developer to create a new component for the integration.

**Test Engineer**
Testing occurs at many levels in an SOA-based solution. Each solution component needs to be tested independently to ensure that it behaves correctly. Then the focus shifts to demonstrating
that the pieces integrate correctly. A large or complex solution requires particular attention to testing if there is to be a smooth transition into production.

- **Test Architect**: The Test Architect specifies how the solution is tested and identifies the additional code required to test pieces of the solution in isolation. The Test Architect can exploit the presence of the service interfaces as a place to insert test code to check that:
  - A component that calls the service interface can respond to all types of data.
  - A component that implements a service interface can process all types of requests.

In this scenario, the Test Architect takes advantage of the Common Customer Data Service to simplify the runtime environment needed to test the new J2EE application. By specifying a J2EE-based implementation of the Common Customer Data Service to manage the test customer data, the new application can be functionally tested entirely within a J2EE test environment, reducing the development cycle on this, the largest part of the new code.

- **Test Implementer**: The Test Implementer writes the additional test code, runs the tests to validate the solution, then reports errors to the appropriate developer and tests the resulting fixes. The Test Implementer is involved at multiple stages of the project, checking the functional correctness of individual components, performing integration testing of groups of components, and, during deployment, testing that the solution is production ready. For example, can it be started, stopped, backed up, and recovered after a system failure as well as maintained?

**Deploy the solution**

When the solution is complete and working from a development point of view, the organization needs to change to adopt and support it. This requires preparation and planning at both the business and IT levels.

**Business Operations Manager**

The Business Operations Manager runs all or part of the business that is adopting the new business offering, including setting up his or her part of the organization to make use of the new solution. Because a service-oriented project typically links parts of the business together, the Business Operations Manager collaborates with other Business Operations Managers to ensure that their procedures are complementary. For this project, the Business Operations Managers check to make sure that they have the same policies on the collection and use of customer data to ensure they don't accidentally violate privacy agreements made with existing customers.

**Resilience Engineer**

The Resilience Engineer is concerned with ensuring that the new solution is available when needed. This includes ensuring availability during peak loads -- changing business campaigns -- and after a failure -- whether it is small, major, or complete.

In a service-oriented environment, a solution needs to be resilient end-to-end. This includes ensuring that the back-end systems can handle the increased load that the new offering places on them through the calls to the Common Customer Data Service. These existing systems also need
to be available whenever the new offering needs it. As such, the Resilience Engineer may need to increase the capacity of the existing systems and alter their operational procedures accordingly.

**IT Administrator**
The IT Administrator configures the IT infrastructure that supports the software solution. In this scenario, IT Administrators configure the IBM WebSphere® Application Server clusters to support the new J2EE application. They set up an IBM DB2® database server for the application and a Service Integration (SI) bus to support the Common Customer Data Service interface and mediation. Finally, they configure the server environment for an adapter that connects to one of the existing back-end systems. This provides the platform on which the Release Deployer adds the solution components.

**Release Deployer**
The Release Deployer configures the components for the solution in the production environment. This includes installing the solution components and configuring them with the resources, such as database tables and queues, that they need to operate. In some circumstances, the Release Deployer needs additional resources, such as user IDs, defined in the existing systems. Typically, the Release Deployer works with the appropriate IT Administrator for that system to get them defined.

When all of the solution is in place, a Test Implementer ensures all is working correctly. The solution is ready for production.

**Run the solution in production**
Many IT departments encourage their staff to specialize on a particular technology or application. However, a service-oriented solution integrates multiple technologies.

In production, success is measured on the end-to-end availability of the solution. As such, the operational procedures are scoped accordingly. A service-oriented solution requires collaborative working between people with technology-based skills and values those with knowledge that spans many technologies. In addition, links with the business become more important. This is illustrated in the following three roles.

**Service Level Manager**
The Service Level Manager agrees on the level of service for the new solution with the Business Operations Manager. The knowledge of the Service Level Manager needs to cover all the technologies used across the solution, including the existing systems supporting the Common Customer Data Service.

**IT Operator**
The IT Operator monitors the systems and performs housekeeping operations, such as backups. When the Common Customer Data Service is in place, the backups of the customer databases in each of the existing systems need to be synchronized to enable a consistent restore of the customer data after a system failure.

**Incident Analyst**
The Incident Analyst investigates unexpected errors and alerts from the IT systems. Tracking down a missing customer data update may require the Incident Analyst to check the diagnostics on the J2EE server and the existing back-end systems.

**Service-oriented governance**

Finally, you need controls to ensure the reuse of IT assets whenever appropriate without disrupting existing operations. The Enterprise Architect has overall accountability for this and delegates some of the work to a specialized role.

**Change Control Manager**
The Change Control Manager is responsible for approving all changes to production systems. This includes new uses of a service even if this doesn't require a change to the implementation, because it affects the load on the service.

The Change Control Manager works with other experts in the IT department to validate that a change will be successful. For example, the Change Control Manager checks with the Resilience Manager to ensure that the new solution doesn't affect the existing back-end systems. The Change Control Manager ensures that the IT Operator can monitor the new solution and that the housekeeping procedures required by the solution are in place.

**Asset Manager**
The Asset Manager is responsible for maintaining the reusable IT assets for an organization. This includes development, maintenance, and making them available to all appropriate projects.

The IT assets from the Common Customer Data Service includes the service interface definition, any data types definitions used on the interface, and the mediation component. The Asset Manager ensures that they were developed to appropriate standards, including documentation, and then stored in the asset repository.

On subsequent projects, the Asset Manager reviews the designs to ensure that no opportunities for reuse are overlooked.

**Summary and conclusions**

The service-oriented approach allows an organization to integrate and reuse its existing IT assets. This changes how the people in the organization interact, encouraging a more consistent approach across the organization. Through user roles, this article characterized the variety of skills and responsibilities found supporting and using IT systems in an organization and how these roles collaborate in a service-oriented environment.

**Acknowledgements**

Many thanks to the members of the IBM SWG Architecture Board workgroup on SWG User Roles (now Common Customer Roles), without whose great work we would not have had the basis for describing the set of SOA user roles presented in this article, and to Marcia Stockton for her continued support of this user roles effort.