Service-oriented enterprise integration: A case analysis

Exploring the steps to performing enterprise integration

Xin Sheng Mao
Ge Jin
Jian Chen

In this article, you explore the basic steps of service-oriented enterprise integration through a simplified but real case that includes business analysis, service modeling, architecture design, and system implementation throughout the entire project life cycle. You also discover the major technologies involved in service-oriented enterprise integration.

Overview

Here’s the scenario: The customer is an airline company which has built its IT system over several decades. The company's core business system was established during the 1970s and 1980s. Most of these applications were developed on IBM mainframe computers, including the travel ticket-booking system running on Transaction Processing Facility (TPF), and the flight coordinating system running on IMS™. Meanwhile, several non-core business systems are built on UNIX™ and a few applications on .NET that work with other core business applications. Some of these applications were developed by an assembly language or COBOL and run on a mainframe and IMS. Others run on UNIX and Oracle. Most use a terminal user interface, but an increasing number uses a Web-based GUI.

During the last few years, the company has put a great deal of effort into enterprise integration and has already set up an information hub as the basis for the integration infrastructure, allowing core-systems functions to be exposed. Many applications, however still use point-to-point integration. This type of integration has considerably improved information sharing among the systems, but the IT architect and developer team realize that will not accommodate future integration.

The following list outlines issues related to the information hub:

- The information hub was built on host technology because most of the core systems are actually mainframe applications. The adoption of host technology can be done by leveraging the information hub in an open standard environment.
• Because of the limited event mechanisms inherent in the information hub, applications are able to recognize those events that are tightly coupled together.
• Currently, business logic for collaboration between systems are generally hard coded, and it's difficult for them to be reused and recomposed quickly to adapt to new business requirements.

To solve these issues in an enterprise integration scenario, the airline company decided to leverage a "ramp control system" as a proof of concept (POC) to explore enterprisewide integration using service-oriented technology. In the remainder of this article, you'll learn about two specific business processes ("ramp coordination" and "ramp control system") for expansion and the implementation steps that IT technical staffs encountered during this integration project. The focus is on how how to use service-oriented technology to solve these problems.

Business environment analysis
In the aviation industry, the term "ramp coordination" refers to the coordination process for all kinds of necessary business activities during the period from flight take-off to landing. In general, a coordinator is assigned for each flight to take responsibility of ramp coordination work. Business activities of ramp coordination include inspecting aircraft parking spot, unloading cargo, loading cargo, and fueling.

Figure 1 shows a business process of ramp coordination, which is comprised of the following activities.

• Retrieve Flight BO: Retrieve main information from coordination process, which is normally a printed worksheet that is delivered to the ramp coordinator (automated process)
• Check Spot: check safety of parking spot (manual activity)
• Check Unloading: check unloading cargo (manual activity)
• Check Loading: check loading cargo (manual activity)
• Check Push back: check push-back event of door close (manual activity)

Figure 1. Ramp Coordination process flow

![Diagram of ramp coordination process flow](image)

Actually, ramp coordination processes vary for different schedules and flight types. The workflow showed in Figure 1 illustrates the coordination procedure of arriving flights which will depart soon.
We call this a "short turn-around" flight. Besides this type of flight, there are also two other kinds of flights, shown in Figure 2 -- "Arrival Only" and "Departure Only." Arrival Only is the flight that departs within the next day of the arrival day; "Departure Only" is the first flight in a day. Most ramp coordination business activities are similar for all these three types of flights. Each type can also be divided into more subcategories by domestic/abroad, long-distance/short-distance, and so on.

**Figure 2. Arrival Only and Departure Only flight**

Many of these processes share a set of business activities, which are used in different manners. In a service-oriented enterprise integration, reusable business services are identified and extracted from the shared activities. Choreography capabilities are provided by the services so that they can aid in composing business services. Because of reusability, these approaches decrease the cost of integration and speed the development time for integration.

**Service modeling**

Component business model (CBM) and Service-Oriented Model and Architecture (SOMA) are two IBM-recommended methodologies to enable the business component model, service model, and process model. (We will not cover service modeling methodology because it is beyond the scope of this article.)

Ultimately, a service model is the major output of service modeling. Figure 3 shows the ramp coordination service model. The two business components related to the process are:

- **Ramp control.** The component that includes all the activities related to ramp control
- **Flight management:** The component for flight information management, including flight schedule and customer information
The following services are exported by these two business components:

- **Retrieve Flight BO**: Exported by the flight management component, to retrieve flight-related data information
- **Ramp Coordination**: Exported by ramp control component, to arrange process of ramp coordination
- **Check Spot**: Exported by ramp control component, to check safety of aircraft parking spot
- **Check Unloading**: Exported by ramp control component, to check status of unloaded cargo
- **Check Loading**: Exported by ramp control component, to check status of cargo to be loaded
- **Check Push Back**: Exported by ramp control component, to check event of the closing of the aircraft door.

After service identification and exposure, we need to figure out how to realize these services in a customer environment. For this case, Retrieve Flight BO is implemented as an information service, Ramp Coordination is implemented as process service using Business Process Execution Language for Web Services (BPEL4WS) technology, and the other four are implemented through their various staff services. Any of these services can be altered when there is a change to the business or IT environment. In our example, we use staff services to implement the Check Push Back service. In the future, this service could be handled by an automatic system. At that time, to adapt to the change, the staff service implementation will be replaced, but the overall process will remain the same. This is a typical example of service replaceability.

**IT environment analysis**

An IT environment analysis is a major approach to investigate the technology used in the customer’s current IT environment. In our case, IT environment analysis is used to support the decision to use the service realization method in a service model. At the same time, it is also an important input for architecture design.

One of important steps in this process includes investigating the as-is IT environment so that it captures existing systems around ramp control, including correlative systems and its interfaces, as well as communication protocols and exchanged data type. **Figure 4** is a simplified view of this kind of IT environment. It depicts interaction between Ramp Coordination process and related systems. Currently, the Ramp Coordination process focuses on four types of existing applications:

- Retrieve crew’s information from crew management system
• Retrieve passengers' information from ticket booking system
• Retrieve cockpit's information from cockpit management system
• Receive event of flight arrival from flight coordination system

**Figure 4. Ramp Coordination process interaction chart**

As Figure 5 illustrates, flight and ticket information is stored in IMS and TPF systems running on a mainframe computer. Although a mainframe has certain capabilities (like MQ and sockets) to integrate with open systems, it is limited by the mainframe itself. So, how to integrate information within the host becomes an important technical problem for the Ramp Coordination project as well as a big issue for the overall enterprise architecture. Here, service-oriented integration gives us an open and effective approach to integrate these existing systems. In this project, the data is aggregated on the hosts and then exported as an information service, for integrating with an open system. Figure 5 shows us the flight and cockpit information from IMS, crew info from Oracle, and passenger info from TPF, and how these are merged into one business object, Flight BO service. The Retrieve Flight BO service offers the method to retrieve this kind of business object. This particular service separates the interface from the complexity of the implementation. The external applications talk to the interface in the front end instead of using the mainframe system in the back end. As long as the service interface is consistent, the applications using the service can still run as they normally do, even if the host in the back end is replaced by an open system.

Universal connectivity to core systems in the mainframe is achieved by an approach aggregates data of host applications into a coarse-grained business object, and you need to export that as an information service. Moreover, the approach enables evolution of your IT systems because of service replaceability.
High-level architecture design

High-level architecture of the Ramp Coordination system, as Figure 6 illustrates, is based on a business model in the requirement and design phase, as well as the as-is IT environment, which uses traditional IT application development principles.

Figure 6. Architecture of Ramp Coordination

The following four points describe the major architecture elements and the relationship between them.

- **Federation Service**: Processing Ramp Coordination requires four types of data from as-is systems. This data is then combined into one business object (Flight BO) in the service modeling phase. In the preceding description the service called Retrieve Flight BO is identified for retrieving information from as-is systems. In fact, it is a federation service. This service merges the data from the crew management, cockpit management, and ticket booking systems. The data from crew info, cockpit info, and passenger info are existing business logic or business data, and they belong to the connectivity on-ramp service. The connection protocols are Java Database Connectivity (JDBC), IMS J2C connector, and sockets. The crew management system is built on an Oracle database, so the crew info can be retrieved from JDBC directly. Since the cockpit management system is based on IMS for S/390®, IBM offers a J2C connector for IMS; the cockpit info was actually obtained from the J2C connector. The
ticket booking system is based on IBM TPF and because of real-time requirements, sockets is the model for connectivity. The service called Retrieve Flight BO is implemented as an EJB and can be accessed by a Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP) binding. The Flight BO is defined as a service data object (SDO) inside the service, Retrieve Flight BO.

- **Event Service**: The ramp coordinator should be notified of flight arrivals before checking the parking spot. The notification is performed by events from the flight operation system, and the service Flight Arrival is a typical event-detection service. This service delivers the event to the message broker through MQ, and the event is distributed to service Check Spot through the publish and subscribe (pub-sub) capability of Java Message Service. Event service is a very important part in this case. Utilizing the common event service of the enterprise service bus (ESB), there is a limit on the amount of existing information that the hub has to overcome. The communication between services is not point-to-point anymore, but is done in a loosely coupled way that enables event service of ESB. The tightly coupled relationship between applications has been mediated greatly through this approach.

- **Process Service**: Ramp Coordination is implemented as a process service and is executed by a BPEL4WS container of WebSphere® Business Integration Server Foundation, which supports the choreography service, the transaction service, and the staff service. Ramp Coordination is invoked using RMI/IIOP, and in the BPEL4WS container, Web Services Invocation Framework (WSIF) is used for accessing services by means of various protocols, which are part of a transport service in ESB. Staff activities of Ramp Coordination are implemented as a staff service and integrated into the process. In this case, the staff service is implemented by a portlet that runs on WebSphere Portal server. The portal server delivery service enables accessibility from PDA devices, so the ramp coordinator can use them to access the system.

- **Transport service in ESB**: The functionality of the remote change management server (RCMS) is exported as service to participate in the coordination process. Here, WSIF takes responsibility of the transport service to enable the process to access these services through SOAP over HTTP.

**Development process**

Although the development phase in a service-oriented enterprise integration is not essentially different from normal application development, it still has many of the specialties you might find in a development role, responsibility, tool, and method. You can see these development items in Table 1.

**Table 1. Role assignment and support tools**

<table>
<thead>
<tr>
<th>Role</th>
<th>Development method or tools</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business analyst</td>
<td>WebSphere Business Integration Modeler, CBM</td>
<td>Process modeling</td>
</tr>
<tr>
<td>Architect</td>
<td>Rational® Software Architect, SOMA</td>
<td>Service modeling using SOMA high-level architecture design and modeling</td>
</tr>
<tr>
<td>Integration developer A</td>
<td>WebSphere Studio Application Developer Integration Edition</td>
<td>Service and process development</td>
</tr>
</tbody>
</table>
Summary

In this article, you evaluated the steps, processes, and relevant technologies that help achieve service-oriented integration within an enterprise environment. The methodologies, architecture, and programming model are evolving; what we reviewed here, however, gives you a solid base to work from. As various technologies mature and gain traction, the implementation and deployment of a service-oriented enterprise integration solution will continue to become simpler and more efficient.

© Copyright IBM Corporation 2006
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)