Implementation of a Global ESB

Dr. Alexander Königs, Thomas Störmer
Schenker AG
Essen

Keywords
ESB, Weblogic, Oracle Service Bus, Service Oriented Architecture, A2A integration, Governance, Design patterns, Best practises

Abstract
Establishing an ESB in a company requires business and technical rules and guidelines for many areas from governance and architecture to implementation. In this article, we explain how Schenker realize their ESB aiming at a reusable and sustainable service-oriented-architecture, describe best practices, and discuss pros and cons of the chosen approach. Thereby, we explain our development and governance process which is based on quality gates in detail. Furthermore, we present our service design guidelines with a closer look at bottom-up vs. top-down approaches. Finally, we outline our common implementation patterns as well as our technical architecture. As a result, we present our way to implement an ESB in order to inspire other users and to start a discussion of different ways to realize a service-oriented-architecture.

Introduction

Schenker are a global logistics provider in the areas of air, ocean, and land transportation as well as contract logistics and supply chain management. Around the globe about 70.000 employees are working for Schenker. The head office is located in Essen.
For historical reasons the IT application landscape of Schenker is very diverse with about 2.000 different applications. Thereby, the majority of applications is country-specific while only a small set are rolled-out as global standards. Schenker are currently consolidating and trying to reduce the number of applications to around 500.
Before start of the ESB initiative back in 2010, B2B and A2A integrations were created either directly or by using an EDI platform in a point-to-point manner. The creation of point-to-point integrations with such a high number of applications led to a very intricate and hard to maintain integration architecture. Furthermore, the EDI platform suffers from poor support for XML messages and does not provide support for synchronous integrations.
In 2010 Schenker started the initiative to establish an additional solution for A2A integrations. The new solution should come with native XML support and support for synchronous integrations. Furthermore, integrations on the new solution must not be created in a point-to-point manner but focus on reusability instead. Schenker did not pursue a real top-down Service-Oriented-Architecture (SOA) strategy. Rather, the reusability was created on a technical level on the fly in a bottom-up manner. Schenker decided to choose the Oracle Service Bus 11g as the product for the new solution.

Development and governance process

Up to now, the development of integrations is in the responsibility of the different applications that want to integrated themselves via the ESB. In order to ensure a high quality we provide the developers with a number of documents that contain rules which features they are allowed to use ("Solution Architecture" document) and how to apply them correctly ("Development Cookbook" document). Furthermore, we defined a governance process ("Governance Process" document) that specifies all steps that are necessary for creating new integrations. This process starts at the requirements specification, followed by the development specification, continues with the development itself, followed by the testing phases, and eventually with the production launch (cf. Fig. 1).

Thereby, each phase is guarded by a Quality Gate that specifies quality criteria which will be checked and must be fulfilled in order to pass the Quality Gate and to continue with the process. At the specification phases we check the specifications for completeness and correctness. At the development phase we check perform automatic and manual code checks and walkthroughs to ensure that all guidelines from our "Solution Architecture" and "Development Cookbook" documents are adhered to. At the testing stages we check whether sufficient test cases (technical as well as functional and load and performance tests) have been performed and documented. Only if all Quality Gates have been passed successfully, the integration is allowed to be put into production.

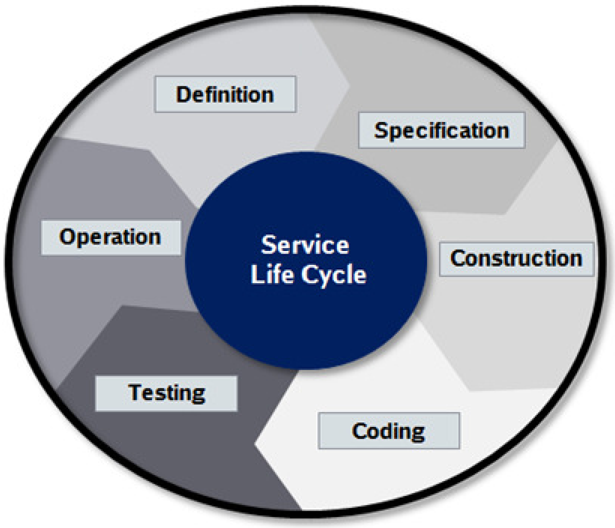

Fig. 1: Overview of the governance process

Service design guidelines

Up to now, Schenker did not pursue a real Service-Oriented Architecture (SOA) strategy for their ESB. Instead of analyzing business processes first, deriving required services from them, and finally implementing generic and reusable services, integrations were created when they were technically required by the applications. However, as soon as the need for an integration was encountered, the idea was to then define a generic and reusable service as best as possible. However, this bottom-up style of service design comes with a lot of drawbacks compared to the pure top-down approach of a real SOA. One main drawback is that as soon as an integration is technically required by applications, the timeframe for implementing generic and reusable services is very limited. This results in suboptimal service designs containing quite a number of compromises. Another drawback is that the generic interface of a reusable service is influence too much by the technical interfaces of the applications rather than driven by abstract business definitions. The main benefit of the bottom-up approach for service design is that only services are implemented that are really necessary. Furthermore, there is no need to make an investment for implementing services upfront that might be used somewhere in the future. However, Schenker have decided in the meantime to aim at real service orientation in the future.

Common implementation patterns

As mentioned above, we provide the integration developers with a "Development Cookbook" document that explains how specific features of the Oracle Service Bus should be utilized for the integrations. One part of this cookbook is a description of a number of design patterns that solve reoccurring problems. Fig. 2 gives an impression of some of our development design patterns.
The "Buffer pattern" is used in situations where System A wants to send a message to System B asynchronously, the message must not be lost, and System A is capable of using unreliable transport

protocols (e.g., http), only. Basically, the idea is that the ESB puts the message received from System A via the unreliable protocol into a JMS message queue as fast as possible. After that the processing of the message can be performed on the reliable JMS transport protocol. Of course, there still is a risk that the message might get lost before it can be put into the message queue but the risk of message loss is minimized by this pattern.
The “Retry pattern” is used in situations where System A wants to send a message to System B, the message must not be lost, and the availability of System B cannot be guaranteed. The basic idea is that the ESB tries to deliver the message to System B. If this fails due to unavailability of System B, the ESB puts the message into a JMS message queue. A proxy service is then listening to this JMS message queue and retries to deliver the message with a specific waiting interval. If System B still is unavailable, the proxy service just rolls-back the message into the JMS message queue for a later retry.
The “Multicast pattern” is used if a message from System A should be forwarded not only to one receiving system but to multiple receivers. The basic idea is to have a proxy service for each receiving system with its own transactional context to deal with each copy of the message independently. The message is then sent to each of these proxy services. Although this pattern works fine, it is not very handy and elegant. We will replace this pattern shortly by using JMS topics instead.

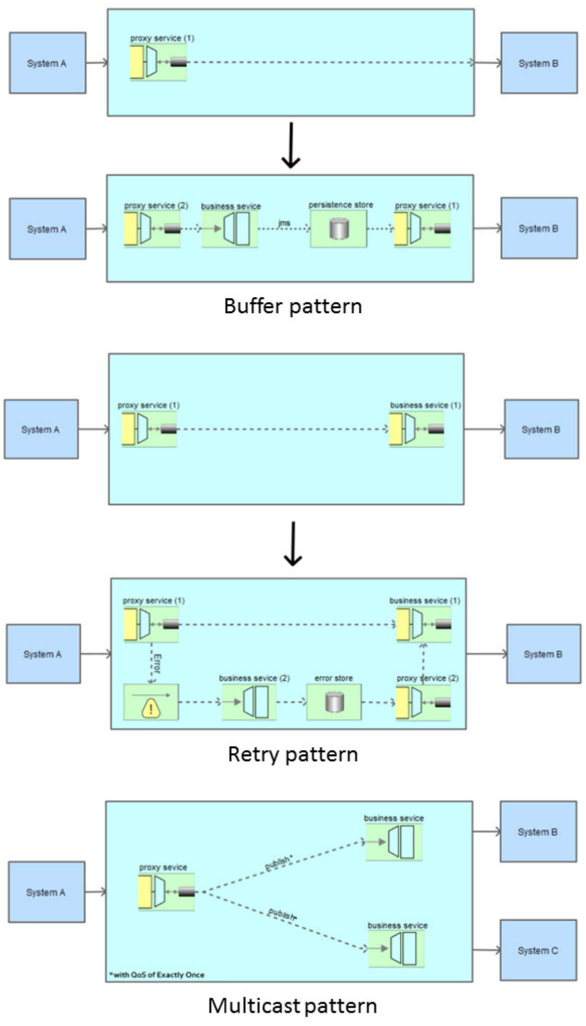

Fig. 2: Examples of development design patterns

Technical architecture

As our ESB is used globally within the Schenker IT for all reasonable application-to-application integrations, we need to provide a highly available integration platform which can easily be scaled as the number of integrations and the number and volume of transferred messages increases over time. At the time of writing there are about 30 applications connected to the ESB exchanging around 100 million messages and 110 GB volume per month.

As described above, we provide four stages of our ESB; i.e., DEV, INT, FAT, and PROD. Thereby, DEV and INT are rather uncritical. We will not describe them in detail here. FAT and PROD are setup identically to allow for meaningful load and performance tests on our FAT environment. Fig. 3 illustrates the technical setup of the PROD stage. This stage consists of four machines that are evenly distributed over two separate data centers that lie about 10 km apart from each other. In each data center one OSB domain. Both OSB domains are running in an active / active setup provided with a shared load balancer. Thereby, each domain contains a cluster of four cluster nodes. In order to scale with the increasing number of integrations, it is easy to extend each domain by additional hardware. As the OSB is stateless and the cluster nodes run independently from each other, linear scaling is possible.

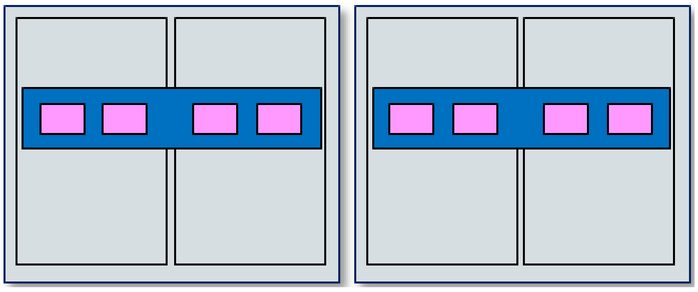

Fig. 3: Technical architecture of PROD stage

Summary

In this article we have outlined how Schenker realized a global ESB by using the Oracle Service Bus. In particular, we presented our development and governance process which is based on quality gates. Furthermore, we discussed the benefit of a top-down service design in contrast to a bottom-up approach. In addition, we presented some of our common implementation patterns that we provide our integration developers with as part of our development cookbook. Finally, we introduced our technical architecture which aims at high-availability and scalability.

Contacts

Dr. Alexander Königs		Thomas Störmer
Schenker AG		Schenker AG
Kruppstr. 4		Kruppstr. 4
D-45128 Essen		D-45128 Essen
Phone:	+49 (0) 201-8781 8141	+49 (0) 201-8781 8441
E-Mail:	alexander.koenigs@dbschenker.com	thomas.stoermer@dbschenker.com
Internet:	http://www.dbschenker.com/	http://www.dbschenker.com/