Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus

Level: Advanced

Jean-Louis Maréchaux (jlmarech@ca.ibm.com), IT Architect, IBM

28 Mar 2006

Today's business applications rarely live in isolation. They need to be connected in order to create an integrated solution from which an organization can derive value. Service-Oriented Architecture (SOA) and Event-Driven Architecture (EDA) are two different paradigms that address complex integration challenges. How can organizations choose the better approach to meet their needs? Actually they don't have to choose: an Enterprise Service Bus (ESB) allows for the implementation of both the SOA and the EDA concepts.

Introduction

To be able to adapt to market changes, organizations tend to focus on flexibility and responsiveness. The IT challenge has usually been to support this business vision with the appropriate architectures and technologies. Early initiatives were to break monolithic applications into callable sub-routines but the advance of remote object invocation and messaging processing changed that.

More recently, it has become crucial to increase the reuse of existing assets in the organization (which increased return on investment) and to assemble heterogeneous applications to form a coherent business solution. This has helped to drive the adoption of SOA and EDA. These two different design paradigms are aimed at maximizing the reuse of application-neutral services that increase IT adaptability and efficiency. But building and deploying large-scale integration solutions has never been easy to achieve. That’s where the ESB comes into play, because it simplifies the realization of flexible and reliable architectures (SOA and EDA) for mission critical applications.

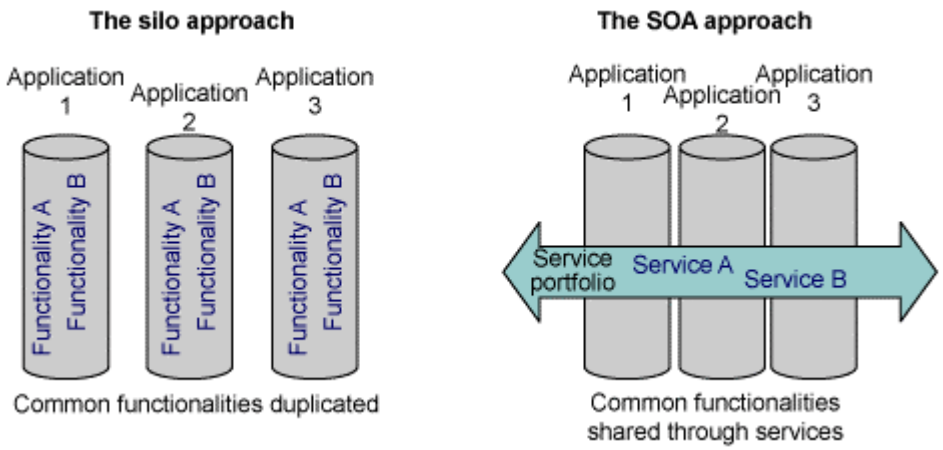
Service-Oriented Architecture

SOA is an architectural concept in which all functions, or services, are defined using a description language and where their interfaces are discoverable over a network. The interface is defined in a neutral manner that is independent of the hardware platform, the operating system, and the programming language in which the service is implemented.

One of the most important advantages of a SOA is the ability to get away from an isolationist practice in software development, where each department builds its own

system without any knowledge of what has already been done by others in the organization. This "silo" approach leads to inefficient and costly situations where the same functionality is developed, deployed and maintained multiple times. A SOA is based on a service portfolio shared across the organization and it provides a way to efficiently reuse and integrate existing assets, as shown in Figure 1:

Figure 1: the "silo" approach versus the SOA approach



SOA is based on a conventional request/reply mechanism, as seen in Figure 2. A service consumer invokes a service provider through the network and has to wait until the completion of the operation on the provider side.

Figure 2: The request/reply mechanism in a SOA

Table 1 summarizes the fundamental characteristics of a SOA solution:

Table 1: Fundamental SOA characteristics

Capability	Description
Loosely coupled interactions	Services are invoked independently of their technology and location
One-to-one communications	One specific service is invoked by one consumer at a time. The communications are bidirectional
Consumer-based trigger	The flow of control is initiated by the client (the service consumer)

Synchronous	Replies are sent back to the consumer in a synchronous way

Event-Driven Architecture

In 2003, Gartner (see Resources) introduced a new terminology to describe a design paradigm based on events: Event-Driven Architecture (EDA). EDA defines a methodology for designing and implementing applications and systems in which events transmit between decoupled software components and services. EDA does not replace, but rather, complements the SOA. While SOA is generally a better fit for a request/response exchange, EDA introduces long-running asynchronous process capabilities. Moreover, an EDA node posts events and does not depend on the availability of a published service. It is really decoupled from the other nodes. EDA is sometimes also referred to as "event-driven SOA".

EDA uses messaging to communicate among two or more application processes. The communication is initiated by an "event". This trigger typically corresponds to some business occurrence. Any subscribers to that event are then notified and thus activated, as shown in Figure 3:

Figure 3. The publish/subscribe mechanism in an Event-Driven Architecture

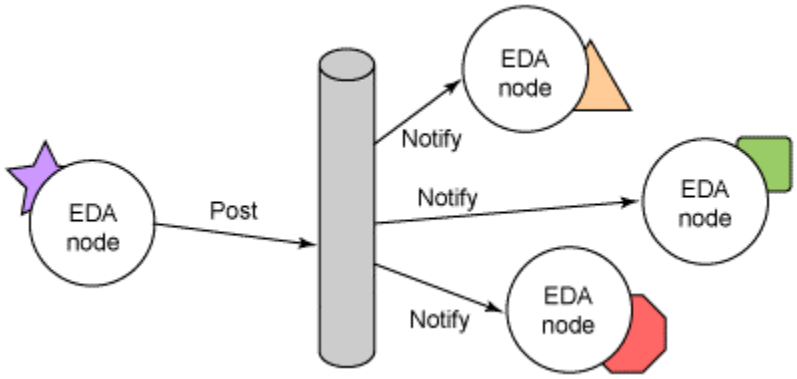


Table 2 summarizes the fundamental characteristics of an EDA:

Table 2: Fundamental EDA characteristics

Capability	Description
Decoupled interactions	Event publishers are not aware of the existence of event subscribers
Many-to-many communications	Publish/Subscribe messaging where one specific event can impact many subscribers
Event-based trigger	Flow of control that is determined by the recipient, based on an event posted
Asynchronous	Supports asynchronous operations through event messaging

Enterprise Service Bus

Definition

An Enterprise Service Bus (ESB) combines event-driven and service oriented approaches to simplify integration of business units, bridging heterogeneous platforms and environments. The ESB acts as an intermediary layer to enable communication between different application processes. A service deployed onto an Enterprise Service Bus can be triggered by a consumer or an event. It supports synchronous and asynchronous, facilitating interactions between one or many stakeholders (one-to-one or many-to-many communications). So the ESB provides all the capabilities of both SOA and EDA paradigms (see Table 1 and Table 2).

An Enterprise Service Bus is an architectural pattern and can be implemented by many different products within the organization, and assembled together to act as a federated bus. More and more vendors are now offering a complete product to fulfill enterprise integration needs. For instance, IBM WebSphere® Enterprise Service Bus (see Resources) delivers an integration bus to connect applications efficiently, leveraging standards like web services and J2EE.

ESB services

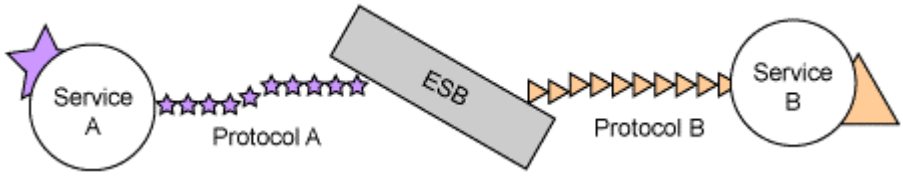
There is no official specification to define what an ESB implementation should be, but it is commonly accepted that it must at least provide *transport*, *event* and *mediation* services to facilitate the integration of large-scale heterogeneous applications.

Transport services must ensure the delivery of messages among the business processes interconnected via the enterprise bus. Transport also includes content-based routing. It means it can direct messages to different destinations. As part of a mission-critical environment, these services are transactional, secured and monitored.

Event services provide event detection, triggering and distribution capabilities. They are related to the notion of event processing, a technique for analyzing and controlling the complex series of interrelated events in Event-Driven Architectures (EDA). Event-driven paradigms are not new. However, they are gaining industry momentum and represent the core concepts of emerging technologies like Complex Event Processing (see Resources).

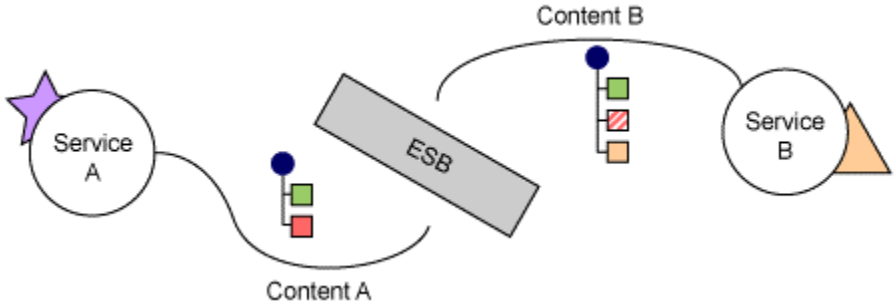
Mediation services address two different purposes. First, the mediation ensures the necessary protocol matching to integrate heterogeneous systems. As two different services do not have to use the same transport protocol, the mediation service takes care of the transformation from one protocol to the other, so that the communication is possible. The protocol switch is transparent for all the participating services of a business transaction.

Figure 4: Protocol mediation – the protocol is transformed by the ESB



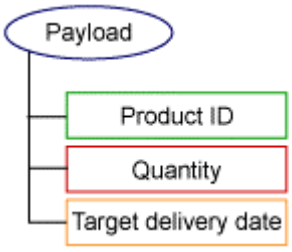
Second, the mediation offers the possibility to transform the content of any message. This is a key service for business integration. It ensures that the data which transits through the bus is understandable by any process. Moreover, the mediation enables content augmentation to enrich a message with any additional information. The content transformation is managed by the bus: it is transparent for any participating service.

Figure 5: Content mediation – the message content is transformed and augmented by the ESB



Let's take an example to illustrate the content mediation benefits. A fictitious company called Yummy Inc. provides online catering services. In order to plan the menus they offer to their customers, they need to verify the availability and the prices of the food items from their supplier. The typical structure of the message they send to obtain this information contains a product identification, a quantity and a target delivery date.

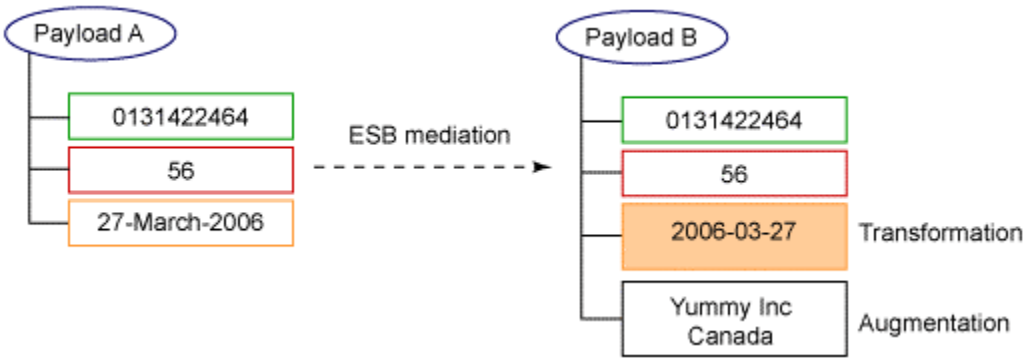
Figure 6: Structure of the availability message



Of course, Yummy Inc. and its supplier do not have the same way to represent the information. For example, the dates are not harmonized in both systems. Moreover, the supplier needs a delivery location because Yummy Inc. is not the only company they

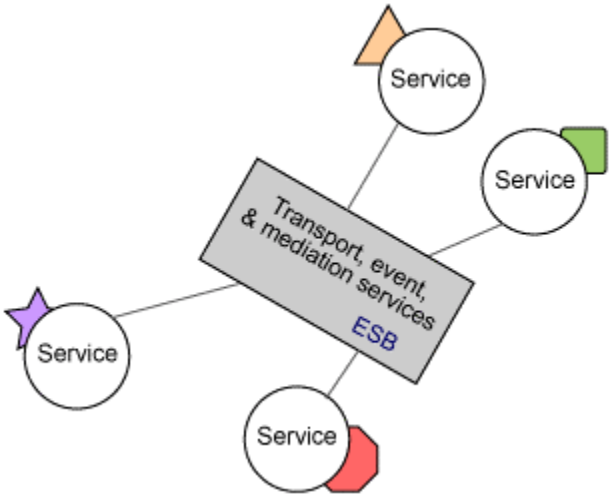
deal with. The ESB mediation service can transform and augment the information of a transiting message so that the target service receive all the information it requires, as illustrated in Figure 7:

Figure 7: Content transformation and augmentation in an ESB mediation



Leveraging the key technical services previously defined, an ESB offers a flexible connectivity infrastructure for integrating loosely-coupled applications. It supports both SOA and EDA paradigms.

Figure 8: Bridging services with an Enterprise Service Bus



ESB benefits

Leveraging its internal services, an ESB solution provides a variety of benefits. In essence, it simplifies the task of connecting dissimilar applications and ultimately improves business agility, and provides the following:

- ***Standard-based connectivity***
  As the integration backbone between many heterogeneous systems, it is

essential for an ESB to provide many different integration techniques, and to leverage a large choice of standard technologies.
The messaging integration usually supports the Java™ Message Service (JMS) API, while the connectivity with enterprise information systems is provided by the J2EE Connector Architecture (JCA). To ensure Web service interoperability, an ESB supports the JAX-RPC programming model. Integration between different ESB components can be standardized by the Java Business Integration specification (JBI) (see Resources).

- ***Pervasive integration***
  An ESB is by nature pervasive because it can integrate applications across different departments, business units or even business partners. Moreover, its core architectural principle is also to facilitate the communication between applications built on heterogeneous development environments. For example, an ESB solution can bridge different programming languages like J2EE, C++ or .Net.

- ***Reliable integration***
  The ESB architectural pattern improves system security, scalability and availability. As it leverages SOA and EDA, the Enterprise Service Bus provides both synchronous and asynchronous capabilities. The transport service ensures reliable delivery and transactional integrity. So every characteristic of an ESB tends to strengthen its robustness, minimizing the risk of failure of the integrated and federated solution.

Conclusion

The Enterprise Service Bus is an architectural pattern that facilitates and simplifies business integration through transport, event and mediation services. It connects and mediates all communications and interactions between heterogeneous nodes, both in a Service-Oriented Architecture (synchronous one-to-one approach) and an Event-Driven Architecture (asynchronous many-to-many approach). An ESB is today's most effective way to address complex integration challenges and is the technical solution that provides the greatest business flexibility and efficient connectivity between dissimilar applications.

Resources

Learn

- Stay current with SOA with developerWorks technical articles.
- Read more about Complex Event Processing.
- Find out how Event-Driven Architecture has been introduced as a new concept by Gartner.
- See specifications for the Java platform:
  - Java Message Service (JSR-914)
  - Java Connector Architecture (JSR-112)
  - Java APIs for XML based RPC (JSR-101)
  - Java Business Integration (JSR-208)

.

Get products and technologies

- Obtain the latest version of IBM WebSphere Enterprise Service BUS, a new product designed to provide an Enterprise Service Bus for IT environments built around open standards and SOA.
- Download a free trial version of IBM WebSphere Application Server. Its default messaging system enables transport, event and mediation services.

Discuss

- Participate in developerWorks blogs and get involved in the developerWorks community.

About the author

Jean-Louis Maréchaux works as an IT Architect for the IBM Business Consulting Services group in Canada. His interests and expertise include J2EE architecture, Web services technologies, SOA, and engineering process (IBM Rational® Unified Process). You can contact him at jlmarech@ca.ibm.com.