Enterprise Service Bus in detail

Lucia Kapová
DISTRIBUTED SYSTEMS RESEARCH GROUP
http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
My last presentation was about …

**Web Process Lifecycle**
- Specification
- Discovery
- Composition
- Execution

- design (composition)
- analysis (verification)
- validation (simulation)

**Diagram**
- Discovery
  - Find Matches
  - Rank Services
  - Select a Service
- Design
  - Create Process WSDL
  - Create Process Template and Add Activities
  - Find Ontologies & Annotate Activity Requirements
  - Add Control Flow
- Composition
  - Add to Process
  - Data Transformation
  - Data Flow
- Execution
  - Generate Process
  - Validate Syntax
  - Execute
Orchestration versus Choreography

Orchestration (coordinator)

Web service 1

1:RECEIVE

5:REPLY

Web service 2

2:INVOKE

Mule ESB

Web service 3

3:INVOKE

Web service N

Web service 4

5:INVOKE

Web service 1

1:INVOKE

Sonic ESB

Web service 3

3:REPLY

Web service 2

2:INVOKE

Web service 1

4:INVOKE

Web service 3
What is ESB?

“An Enterprise Service Bus (ESB) is software infrastructure that enables SOA by acting as an intermediary layer of middleware through which a set of reusable business services are made widely available.”

Source: Mike Gilpin, Forrester Research, August 2004
Enterprise Integration Patterns

- based on a set of layered assets
- any existing development methodology

Customer requirements ➔ Composite patterns ➔ Business patterns ➔ Integration patterns ➔ Application patterns ➔ Runtime patterns ➔ Product mappings

ESB pattern
Application patterns

- high level logical components and their interactions
- break the application down into the most basic conceptual components

Diagram:
- Serial Process
  - Variation: Serial Workflow
  - Source Application
  - Serial Process Rules Tier
  - Target Application
- Parallel Process
  - Variation: Parallel Workflow
  - Source Application
  - Parallel Process Rules Tier
  - Target Application
- Direct Connection
  - Variations: Message/Call Connection
  - Source Application
  - Connection Rules
  - Target Application
- Broker
  - Variation: Router
  - Source Application
  - Broker Rules
  - Target Application
Broker Application pattern

- based on a 1-N topology
- Allows a single interaction from the source application to be distributed to multiple target applications
Router Application pattern

- variation of the Broker application pattern
- source application initiates an interaction with at most one of multiple target applications
Runtime patterns

- more explicit functions to the Application pattern
- There are two categories of Runtime patterns:
  - Generic
  - SOA profile

- The SOA profile category of Pattern apply specifically to service-oriented architecture solutions
Enterprise Service Bus pattern

- Broker and Router application patterns map to the Enterprise Service Bus runtime pattern
Enterprise Service Bus pattern – level 1
Enterprise Service Bus pattern - nodes

• **App Service / Services**
  - Requested and provided services

• **Hub**
  - ESB functions
  - Fundamental service integration role
  - Multiple integration styles

• **Namespace Directory**
  - provides routing information

• **Administration and Security Services**
  - administration infrastructure and security policy

• **Business Service Directory**
  - details of available services
How it all works?

Inside the ESB

- J2EE™ APPLICATION
- PACKAGED APPLICATION & LEGACY SYSTEMS
- .NET™ APPLICATION

- PARTNER SYSTEM
- WEB SERVICE

- SERVICES
- SERVICE CONTAINER
- MEDIATION LAYER
- COMMUNICATION BACKBONE
- NETWORK
Lightweight Service Container

Services can be independently scaled

Parallel processing/load balancing

A way to host and control services across platforms
ESB endpoint

ESB endpoint is event-driven
Behavior controlled through configuration not code

Configurable Endpoints
Workflow Supported Exception Processing (WSEP)
Service implementation

Public void service (ESBServiceContect ctx) throws ESBServiceException {

    //get any runtime process parameters
    Parameters params = ctx.getParameters();

    //get the message
    ESBEnvelope env = null;
    env = ctx.getNextIncoming();

    if (env != null) {
        ESBMessage msg = env.getMessage();
        //operate on the message
    }

    //put the message to the Exit Endpoint
    ctx.addOutgoing(env);
}
Lucia Kapová
Seminar Series 2006

Services

Service composition:
1. Based on itinerary
2. Content based
Inside MULE ESB implementation

- First open-source messaging framework based on Enterprise Service Bus pattern
Mule ESB introduction
Architecture

Mule Manager

IoC Container
Spring/Pico/Plexus

Security Manager

Transaction Manager

Notification Manager

Agents
Jmx Agent
Mule Admin Agent
Discovery Agent

Mule Model (SEDA)
Managed Components

Mule Transport
Xml Transformers
Object Transformers
Message Routers
Filters
Transport Providers
Jms Soap Http Multicast Tcp Vm File Jdbc Xmpp Pop3 Smtp ...
Transport providers
**UMO components**

- **Default Constructor**
- **initialise**
  - org.mule.umo.lifecyclesimalisable
- **start**
  - org.mule.umo.lifecyclescaltable
- **onCall**
  - org.mule.umo.lifecyclescaltable
- **stop**
  - org.mule.umo.lifecyclescaltable
- **dispose**
  - org.mule.umo.lifecyclescaltable

- **Called when the component is resolver by the componentResolver**
- **Called when the component is registered with the Mule Model**
- **Called when the server is started**
- **Called when an event is received for the component. By configuring an EntryPointResolver this method can be resolved to any method on your object**
- **Called when the Mule server is stopped. Bear in mind the server can be started again after stop is called**
- **Called when the Mule server is shutdown. Clear up any resources here**

- **Email**
- **SOAP**
- **JDBC**
- **File**
Transformers

- JMS Text Message To String
- Encrypted String To String
- XML to Bean
- String to Email Message
- XSLT Transform To HTML
- Bean To XML
Routers

```
<inbound-router>
  <catch-all-strategy className="org.mule.tck.testmodels.mule.TestCatchAllStrategy"/>
  <router className="org.mule.routing.inbound.SelectiveConsumer">
    <filter expectedType="java.lang.String" className="org.mule.routing.filters.PayloadTypeFilter"/>
  </router>
  <router className="org.mule.routing.inbound.Aggregator"/>
</inbound-router>

<outbound-router>
  <catch-all-strategy className="org.mulerouting.ForwardingCatchAllStrategy" provider="catchAll"/>
  <router providers="TestApple-Out"
    className="org.mule.routing.outbound.FilteringOutboundRouter">
    <filter expectedType="java.lang.String" className="org.mule.routing.filters.PayloadTypeFilter"/>
  </router>
  <router providers="waterMelonProvider"
    className="org.mule.routing.outbound.FilteringOutboundRouter">
    <filter className="org.mule.routing.filters.logic.AndFilter">
      <left-filter pattern="the quick brown (.*)" className="org.mule.routing.filters.RegExFilter"/>
      <right-filter pattern="(.*) brown (.*)" className="org.mule.routing.filters.RegExFilter"/>
    </filter>
  </router>
</outbound-router>
```
Interceptors

Diagram showing the flow of data through various components such as Inbound Router, Logger, Profiler, Permission Check, UMO, and Outbound Router, with interceptors at each step.
<mule-descriptor name="pxe" implementation="org.mule.extras.pxe.PxeComponent">
  <inbound-router>
    <endpoint address="jms://bpel.queue"/>
  </inbound-router>
  <properties>
    <property name="pxeEndpoint" value="com.bptest.process.ProcessSVC/ProcessPORT"/>
    <property name="defaultOperation" value="com.bptest.process:Run"/>
    <property name="configuration" value="async-pxe-sdb"/>
  </properties>
</mule-descriptor>
Questions?

THANK YOU