# Middleware and Web Services

## Lecture 8: Enterprise Service Bus

**doc. Ing. Tomáš Vitvar, Ph.D.**
tomas@vitvar.com • @TomasVitvar • http://vitvar.com

Czech Technical University in Prague
Faculty of Information Technologies • Software and Web Engineering • http://vitvar.com/courses/mdw


Modified: Sun Dec 03 2017, 12:50:47
Humla v0.3

# Overview

- Central intermediary in SOA
  - *Types of services: shared and infrastructure*
  - *Types of processes: Technical and Business*
- ESB Application
  - *Application running on an application server*
  - *Exposes functionality via Web service interface*
  - *Allows to communicate with various messaging protocols*
- Integration Patterns
  - *Technical-level interoperability – message broker*
  - *Location transparency*
  - *Dynamic routing*
  - *Data transformations – mediator*
  - *Resequencing of messages*
  - *Session pooling*
  - *Service orchestrations – BPMN, BPEL*
  - *Message enrichment*

# ESB Vendors

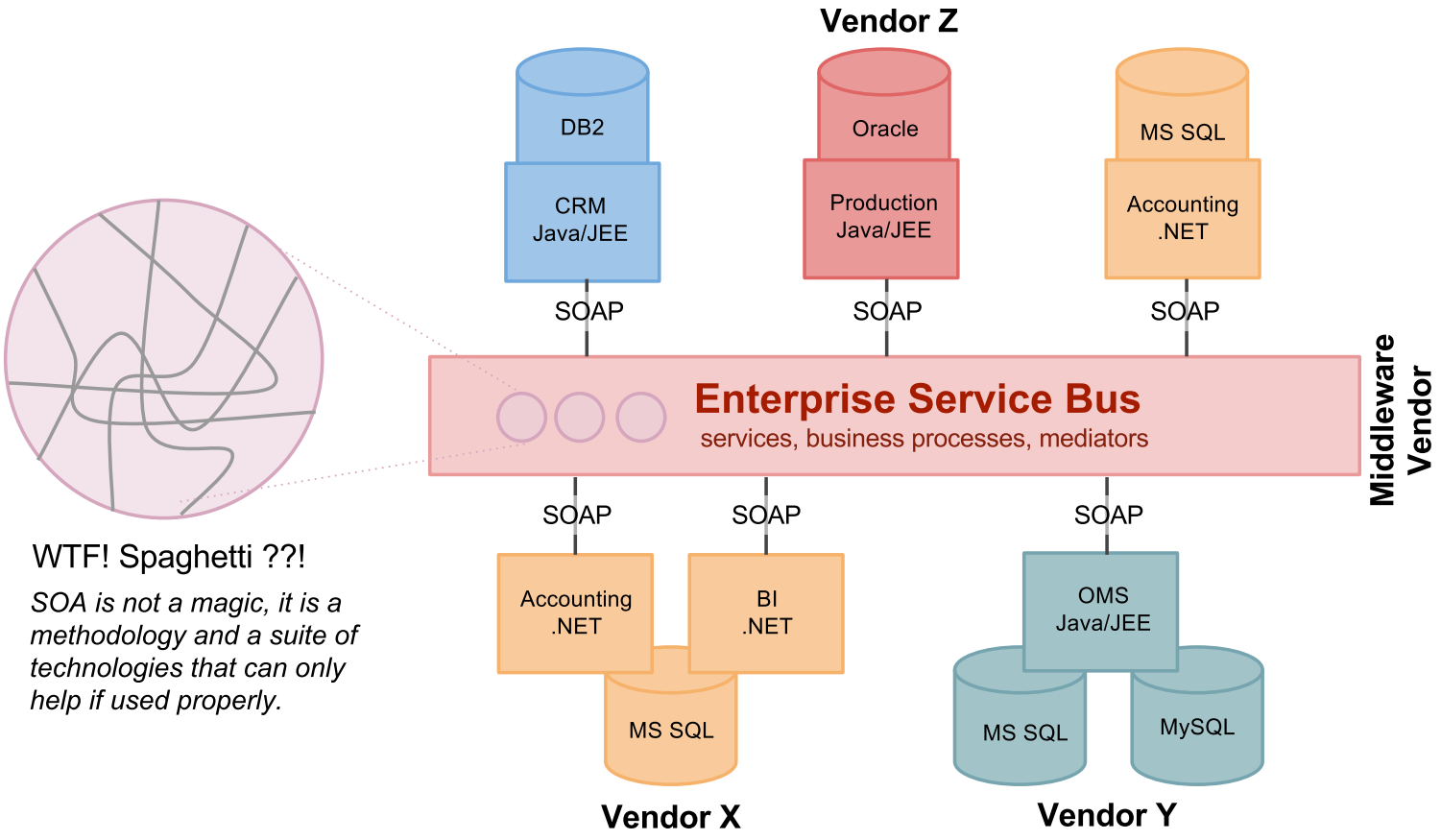
- Oracle
  - *Oracle Service Bus (OSB)*
  - *Oracle SOA Suite*
  - *Oracle Enterprise Gateway (OEG)*
- IBM
  - *IBM WebSphere*
- SAP
  - *SAP NetWeaver*
- Microsoft
  - *.NET Framework*
  - *BizTalk server*
- Opensource
  - *JBoss*
  - *Apache ServiceMix*
  - *WSMX – Semantic Web Service Execution Environment*

# Overview

- Architecture
  - *Service Component Architecture*
  - *Metadata Repository*
  - *Service Types*
- Integration Patterns

# Enterprise Service Bus

- Integration organized
  - *Enterprise Service Bus, to be used wisely*




WTF! Spaghetti ??!

*SOA is not a magic, it is a methodology and a suite of technologies that can only help if used properly.*

# Overview

- Architecture
  - *Service Component Architecture*
  - *Metadata Repository*
  - *Service Types*
- Integration Patterns

# Service Component Architecture

- Industry standard
  - *SCA defines an architecture and a technology for composing applications following SOA principles*
  - *Many adopters: Apache Tuscany, Service Conduit, Oracle SOA Suite 11g*
- SCA Application

***Composite***

*collection of components, services, references*

***Component***

*application bulding block that provides certain functionality; it can be implemented by various technologies (BPEL, Java, etc.)*

***Services***

*exposed services by the application*

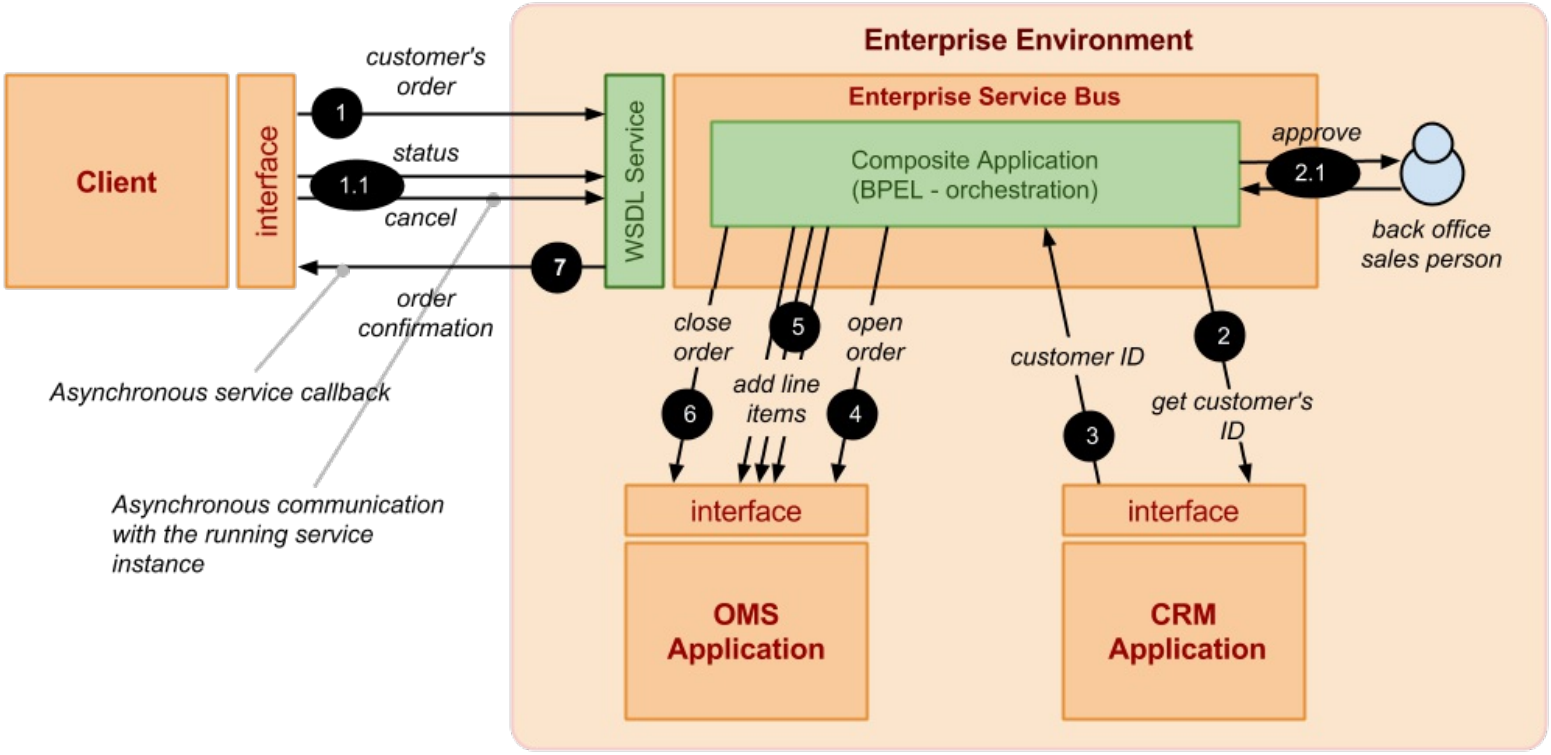
***References***

*references to external services that the application uses*

***Wires***

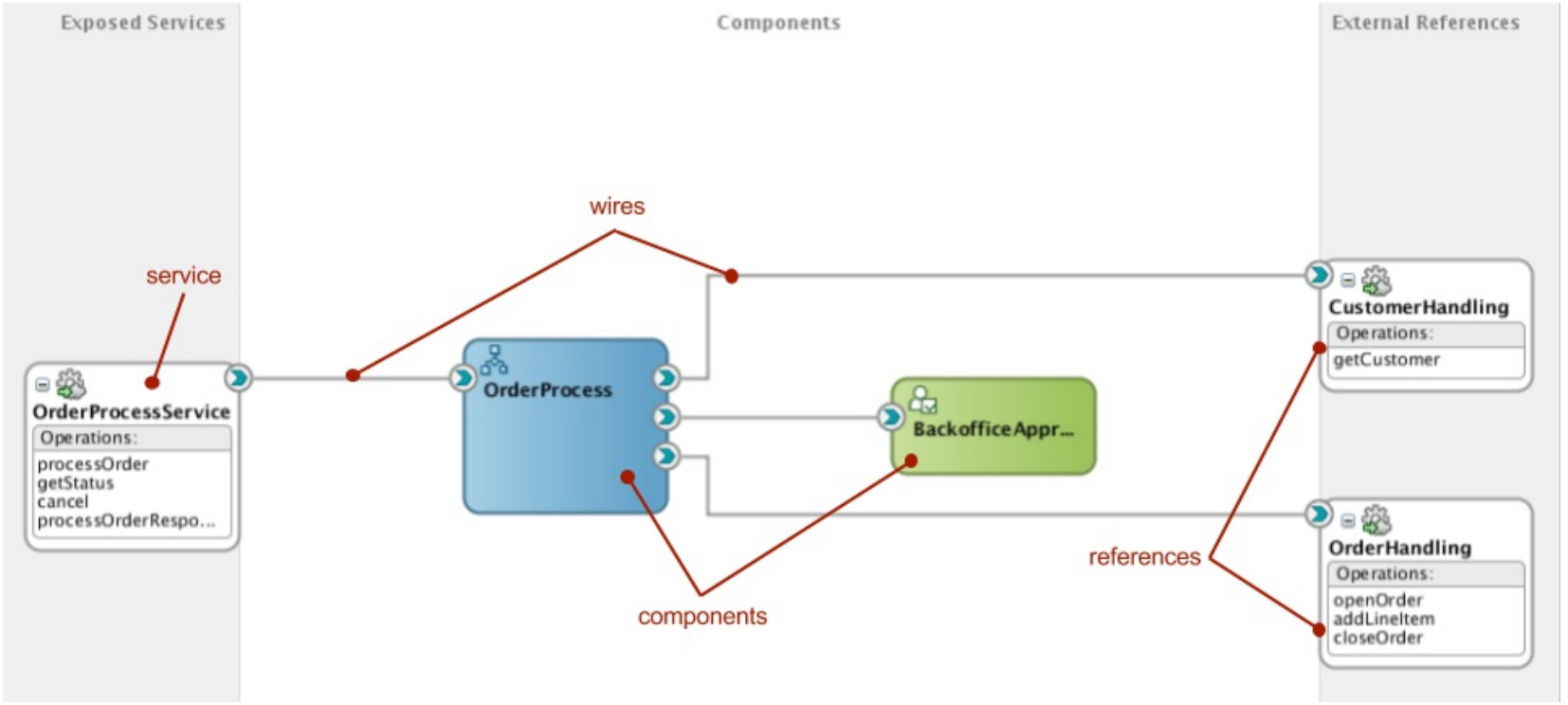
# Simplified Order Process

- Example from Lecture 9



# Order Process – SCA Application

- SCA Application Composite
  - *Service implements Order Process WSDL interface (processOrder, getStatus, cancel, processOrderResponse callback)*
  - *A screenshot from JDeveloper IDE:*



# Composite.xml

- Main configuration file of the composite application
  - *previous slide shows its graphical representation*
- `service` – exposes the composite as a service

`{ns-path}='mimdw.fit.cvut.cz/mdw_examples/ProcessOrder'`

```
1  <service name="OrderProcessService" ui:wsdlLocation="OrderProcess.wsdl">
2    <interface.wsdl
3      interface="http://{ns_path}/OrderProcess#wsdl.interface(OrderProcess)"
4      callbackInterface="http://{ns_path}/OrderProcess#wsdl.interface(OrderProcessCallback)"
5    <binding.ws
6        port="http://{ns-path}/OrderProcess#wsdl.endpoint(OrderProcessService/OrderProcess_p
7      <property name="weblogic.wsee.wsat.transaction.flowOption"
8                type="xs:string" many="false">NEVER</property>
9    </binding.ws>
10   <callback>
11     <binding.ws
12       port="http://{ns-path}/OrderProcess#wsdl.endpoint(OrderProcessService/OrderProcessCa
13   </callback>
14 </service>
```

- `component` – implements the composite in a specific technology

```
1  <component name="OrderProcess" version="2.0">
2    <implementation.bpel src="OrderProcess.bpel"/>
3    <property name="bpel.config.oneWayDeliveryPolicy" type="xs:string"
4              many="false">async.persist</property>
5  </component>
```

# Composite.xml (cont.)

- `reference` – provides an access to an external service

  `{ns-path}='mimdw.fit.cvut.cz/mdw_examples/APP_CRM_GetCustomer'`

```
1  <reference name="CustomerHandling"
2      ui:wsdlLocation="http://sb.vitvar.com/soa-infra/services/mdw-examples/APP_CRM_GetCust
3    <interface.wsdl
4        interface="http://{ns-path}/GetCustomer#wsdl.interface(GetCustomer)"/>
5    <binding.ws
6        port="http://{ns-path}/GetCustomer#wsdl.endpoint(getcustomer_client_ep/GetCustomer_
7        location="http://sb.vitvar.com/soa-infra/services/mdw-examples/APP_CRM_GetCustomer/
8        soapVersion="1.1">
9      <property name="weblogic.wsee.wsat.transaction.flowOption"
10                type="xs:string" many="false">WSDLDriven</property>
11   </binding.ws>
12 </reference>
```

# Order Process SCA Application Instance

# Overview

- Architecture
  - *Service Component Architecture*
  - *Metadata Repository*
  - *Service Types*
- Integration Patterns

# Metadata Repository

- Central Store
  - *Central store for common artefacts used by applications*
- Artefacts
  - *Abstract WSDLs – common interface for integration between clients and ESB and among applications running in ESB*
  - *XML Schemas – common information models used in WSDLs*
    *→ Common Data Model (CDM)*
- Oracle SOA Suite 11g
  - *MDS – Metadata Store; can be in the DB or on file system*
  - *Common artefacts as above + deployed composites*
  - *Artefacts can be referenced/access by* `oramds` *protocol:*

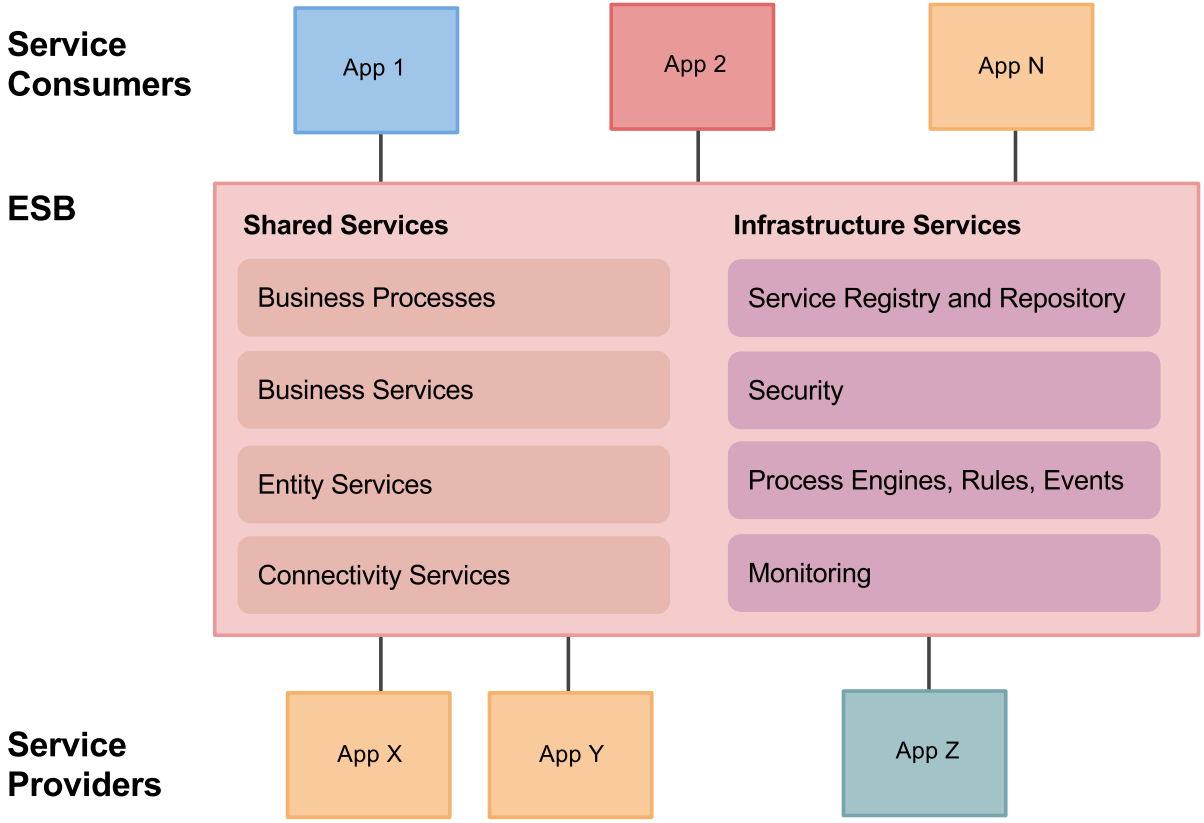
```
1  ...
2  <wsdl:types>
3      <schema
4          xmlns="http://www.w3.org/2001/XMLSchema">
5          <import namespace="http://mimdw.fit.cvut.cz/mdw-examples/cdm/order"
6              schemaLocation="oramds:/apps/MDWMetaData/order.xsd"/>
```

# Overview

- Architecture
  - *Service Component Architecture*
  - *Metadata Repository*
  - *Service Types*
- Integration Patterns

# Service Types

- ESB services
  - *shared services – created for particular domain*
  - *infrastructure services – support integration and interoperability*




*Service and non-service enabled providers*

# Connectivity Services

- Purpose
  - *Adapters for various back-end technologies*
  - *Connectivity to legacy applications*
  - *No business logic, Usually stateless, ESB internal*
- Example
  - *Database adapters*

    → *SQL statement:*

```
1  SELECT ID, NAME FROM CUSTOMERS C
2  WHERE  C.REVENUE > :revenue
```

    `Revenue` – *input parameter*

    `ID, NAME` – *structure of output message*

    → *Expose the SQL statement as a connectivity service*
  - *Example implementation: OSB Proxy service, JCA adapters*

# JCA Adapters

- JCA – Java EE Connector Architecture
  - *Standard Java interface to connect to back-end systems*
  - *Standardized in JSR 112*
  - *Main JCA Adapters*
    - → *JCA DB Adapter – access to DB objects*
    - → *JCA JMS Adapter – JMS queues*
    - → *JCA AQ Adapter – Oracle AQ (in a database)*
    - → *JCA MQ Adapter – IBM MQ*
    - → *JCA FTP Adapter – FTP access*
    - → *JCA File Adapter – File system access*
- Major Features (Contracts)
  - *Connection pool*
    - → *cache of connections to a back-end system (DB, etc.)*
  - *Transaction management*
    - → *JCA adapters can participate in a distributed transaction*

# Entity Services

- Purpose
  - *Expose services on top of one or more entities in a database*
  - *Do not add any specific logic to entities' operations*
    - → *Provide CRUD operations only*
  - *May be used to facilitate a Common Data Model*
    - → *Business entities – entities of CDM*
    - → *Business objects – instances of business entities*
    - → *Business Entity Service – manipulations for business entities*
  - *No business logic, usually stateless, ESB internal*
- Example
  - *Two entities in a database:* `CUSTOMERS`, `ADDRESS` *(1:N)*
  - *Business entity* `CUSTOMER`

```
<customer>
    <name>Company.cz</name>
    <invoice-address>
        …
    </invoice-address>
    <main-address>
        …
    </main-address>
</customer>
```

  - *Operarions: read, write*

# Business Services

- Purpose
  - *Business/integration logic, can be stateful or stateless*
  - *Atomic business activities*
    - *→ direct mapping to back-end application services*
  - *Can be "imported" in ESB to be used in a business process*
  - *Can be exposed by ESB and add values in terms of business/integration logic or technical processing*
- Example
  - *Data transformation*
    - *→ Back-end application service exposed in CDM language*
  - *Message enrichment*
    - *→ Adds information to content from other sources*
  - *Monitoring*
    - *→ Every invocation of the service logged*
    - *→ Monitoring of business metrics*
      - *→ Number of orders, total revenue per customer*

# Business Processes

- Purpose
  - *Business/integration logic, usually stateful*
  - *Complex processes involving invocations of multiple business services at various back-end applications*
  - *Handles transformations from various data formats of back-end applications*
  - *Handles **key-mapping***
    - *→ Business entities exist in multiple systems*
    - *→ Each back-end application maintains its own ID for corresponding business objects*
  - *Usually implemented in a process language such as BPMN or BPEL*
  - *OSB uses its own orchestration language which translates to XQuery*
- Example
  - *Order processing*
    - *→ Get customer information from the CRM system*
    - *→ Add line items to OMS*

# Overview

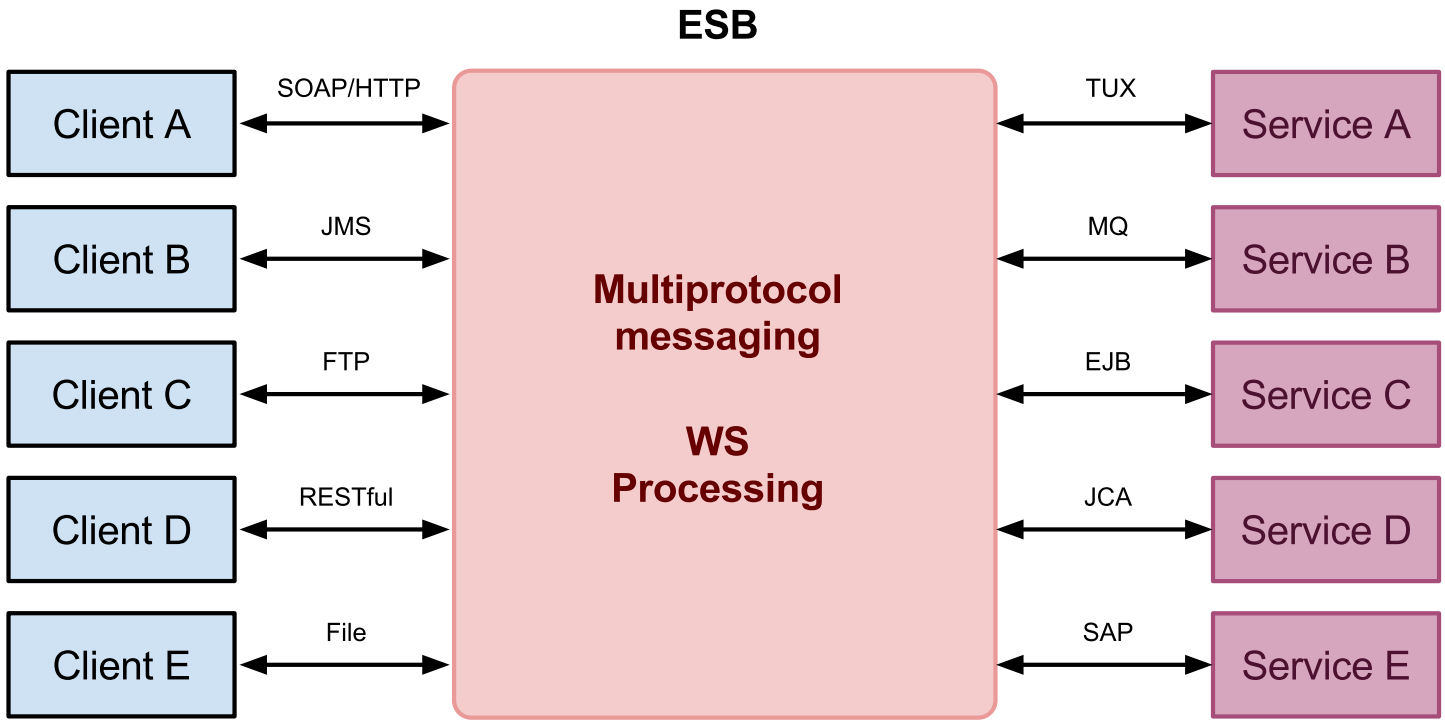
- Architecture
- Integration Patterns

# Overview

- Applied in implementation of business services and processes
  - *Usually a combination of more patterns*
- Technical patterns
  - *Deals with technical aspects of service communication*
  - *Message broker – technical-level interoperability*
  - *Location transparency*
  - *Session pooling*
- Business patterns
  - *Deals with business aspects (message content) of service communication*
  - *Dynamic routing*
  - *Data transformations – mediator*
  - *Service orchestrations – BPMN, BPEL*
  - *Message enrichment*
  - *Resequencing of messages*

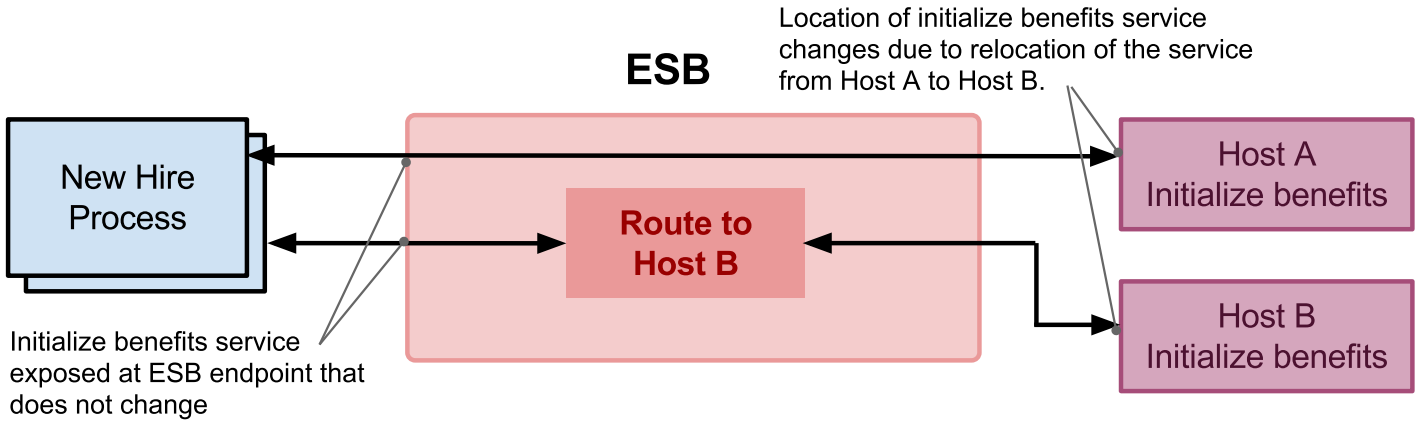
# Message Broker

- Message broker
  - *ESB can mix and match transports both standard and proprietary*



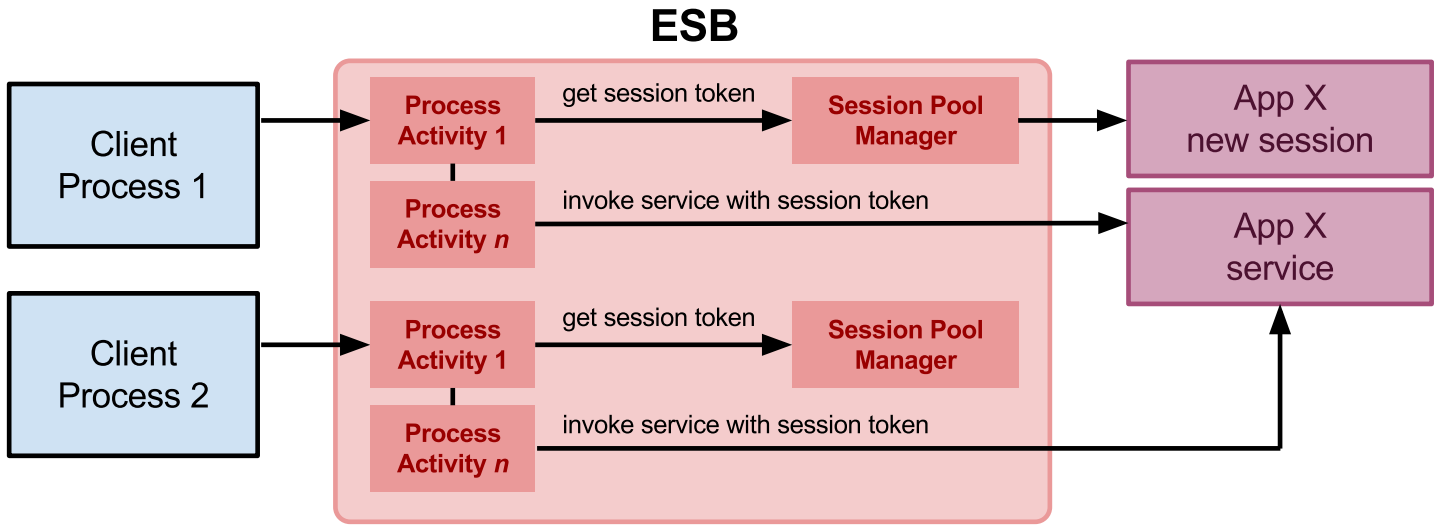
# Location Transparency

- Location transparency
  - *ESB can hide changes in location of services*
  - *Such changes will not affect clients*
  - *Can also be used for load balancing for multiple service instances*



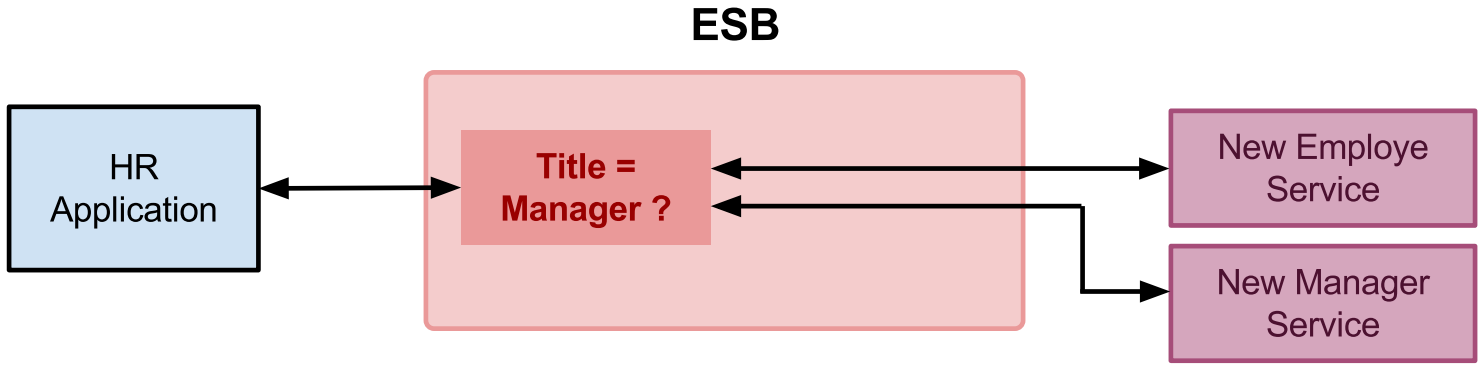
# Session Pooling

- Session Pooling
  - *ESB can maintain a pool of connections (session tokens) to a back-end app when creating a new connection is expensive*
  - *A single session token can be reused by multiple instances of business processes*



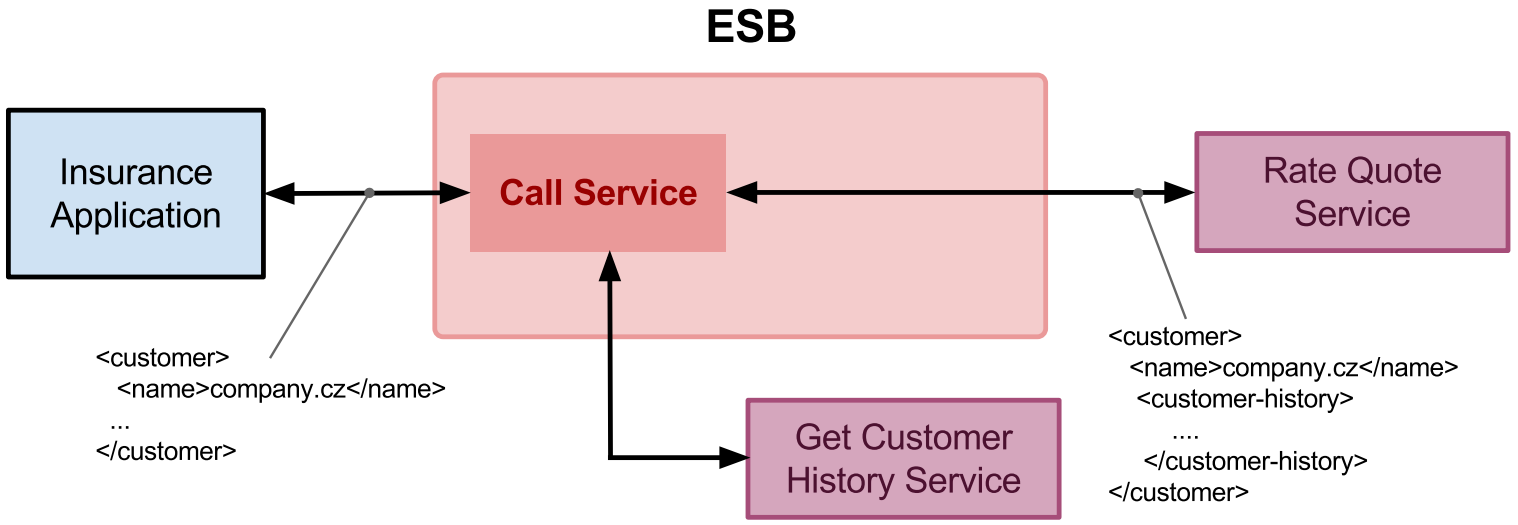
# Dynamic Routing

- Dynamic routing
  - *ESB exposes a service that routes to various back-end services based on message contents.*



# Message Enrichment

- Message enrichmenet
  - *Enriches a message before invoking back-end application service.*

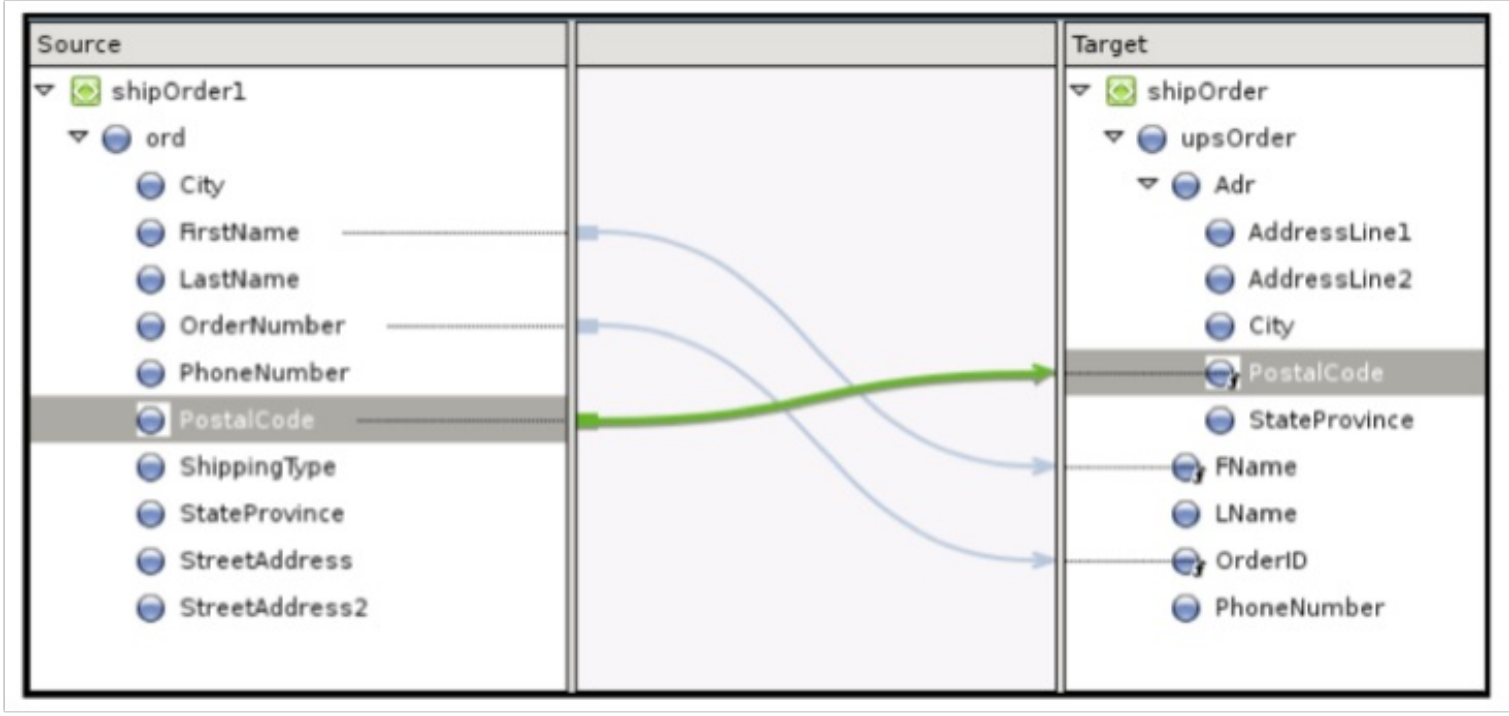


# Data Transformation

- Data transformation phases:
  - *Definition of mapping and execution of mappings*
- Definition of mappings (design-time)
  - *A mapping associates one data structure to another data structure and defines a conversion between them.*
  - *Mapping languages*
    *→ graphical for design that translates to XSLT, XQuery*
    *→ Sometimes implemented in 3rd gen. languages (e.g., Java)*
- Execution of mappings (runtime)
  - *application of mappings to instance data*
- CDM terminology
  - *Application Business Message – back-end app format*
  - *Enterprise Business Message – CDM format*

# Definitions of Data Mapping Example

- Source and target schemas
  - *Source: Order – flat data structure*
  - *Target: UPS order with address as a sub-entity*
  - *Differences in names of entities*
  - *Conversion function applied to postal code*

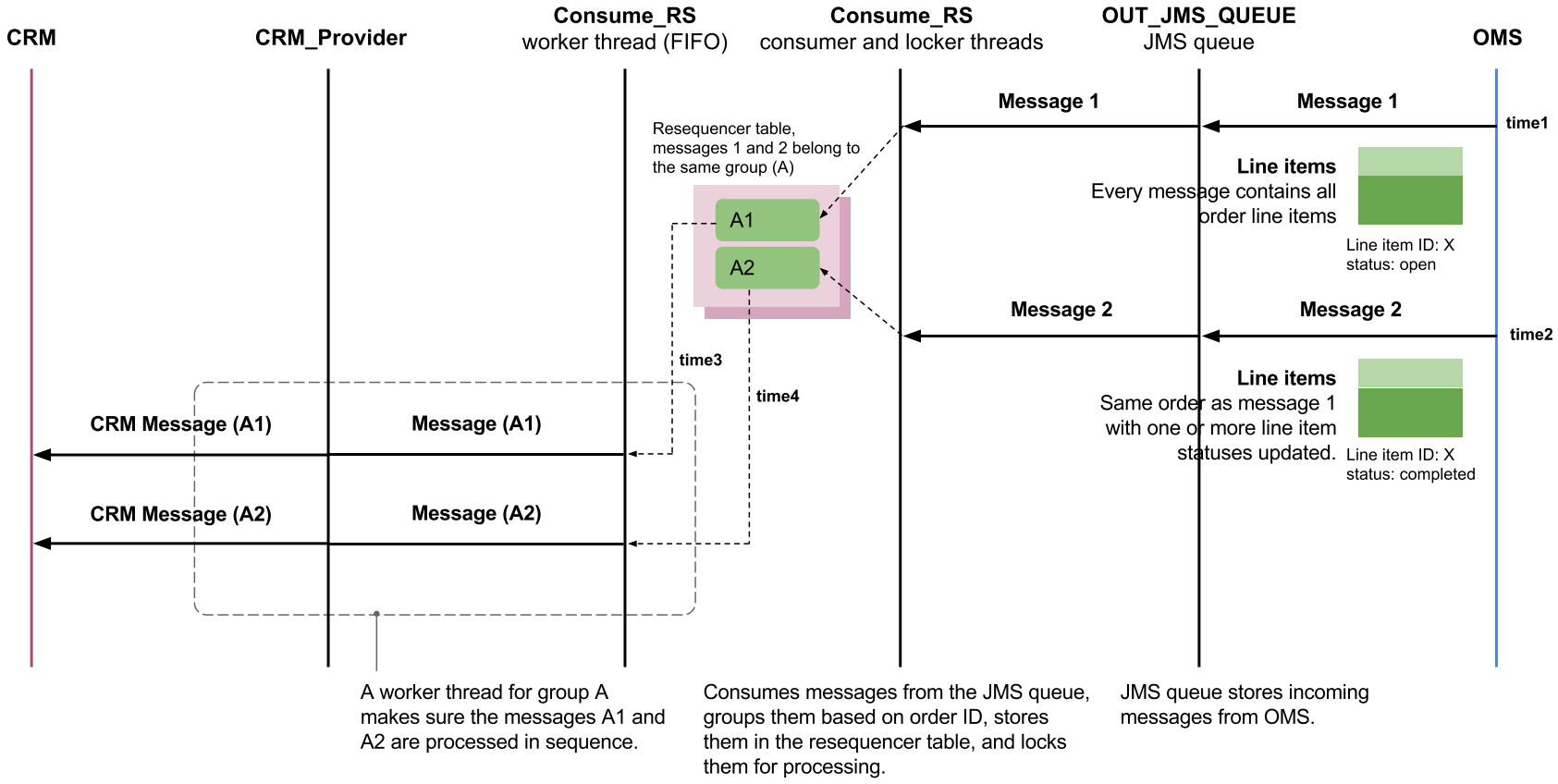


# Service Orchestration

- Orchestration of multiple business services
  - *Includes transformation, message enrichmenet, service callouts, etc.*
  - *A step in orchestration is an activity*
- Patterns
  - *Sequential processing of activities*
  - *Parallel processing of activities with synchronization points*
  - *Decision branches, iterations*
- Technologies
  - *Graphical languages*
  - *Standard representations: BPEL, BPMN*
  - *Proprietary, for example OSB uses graphical language that translates to XQuery*
- Good design
  - *Orchestration faciliates communication in CDM*
  - *Orchestration handles key-mapping*

# Message Sequencing

- Resequencer in update sales order
  - *Every order line item needs to update its status several times (e.g. open, completed)*
  - *Resequencer makes sure that the update status messages arrive to CRM in the same order as they were created in OMS system (FIFO resequencer)*



# Message Aggregation