



the  
**POWER**  
of  
**JAVA™**



JavaOne  
Part of the Oracle and Sun Microsystems

# Building a service with BPEL and Java EE: How Composite Apps and JBI simplify SOA development

Ron Ten-Hove , Peter Walker and Gopalan Raj

Senior Staff Engineers

Sun Microsystems, Inc. [java.sun.com/integration](http://java.sun.com/integration)

TS-3175



# Goal

This session describes how Java EE developers can create composite applications using BPEL using current Java EE tools and application servers that support JBI



# Agenda

Why Services

Why Composite Applications

BPEL in the mix

A Java EE Composite Application

Summary



# Agenda

Why Services

Why Composite Applications

BPEL in the mix

A Java EE Composite Application

Summary

# Why Services?

SOA = an architectural principle for structuring systems

SOA emphasizes the de-coupling of system components

New services are created from existing ones in a synergistic fashion

Strong service definitions are critical

Services can be subsequently re-composed in response to changing business requirements

# What are Services?

A function accessed using XML message exchange

Message exchanges have well known exchange patterns

Services are self-describing, using metadata (WSDL)

# What Does a Service Do?

Transform data

Route messages

Query databases

Orchestrate conversations

Apply business logic

Apply business policy

Handle business exceptions

Solicit approvals

...



# How is a service implemented?

XSLT

EJB

BPEL

SQL

XQuery

Routing Table

Business Rules

EDI Transform

...





# Agenda

Why Services

Why Composite Applications

BPEL in the mix

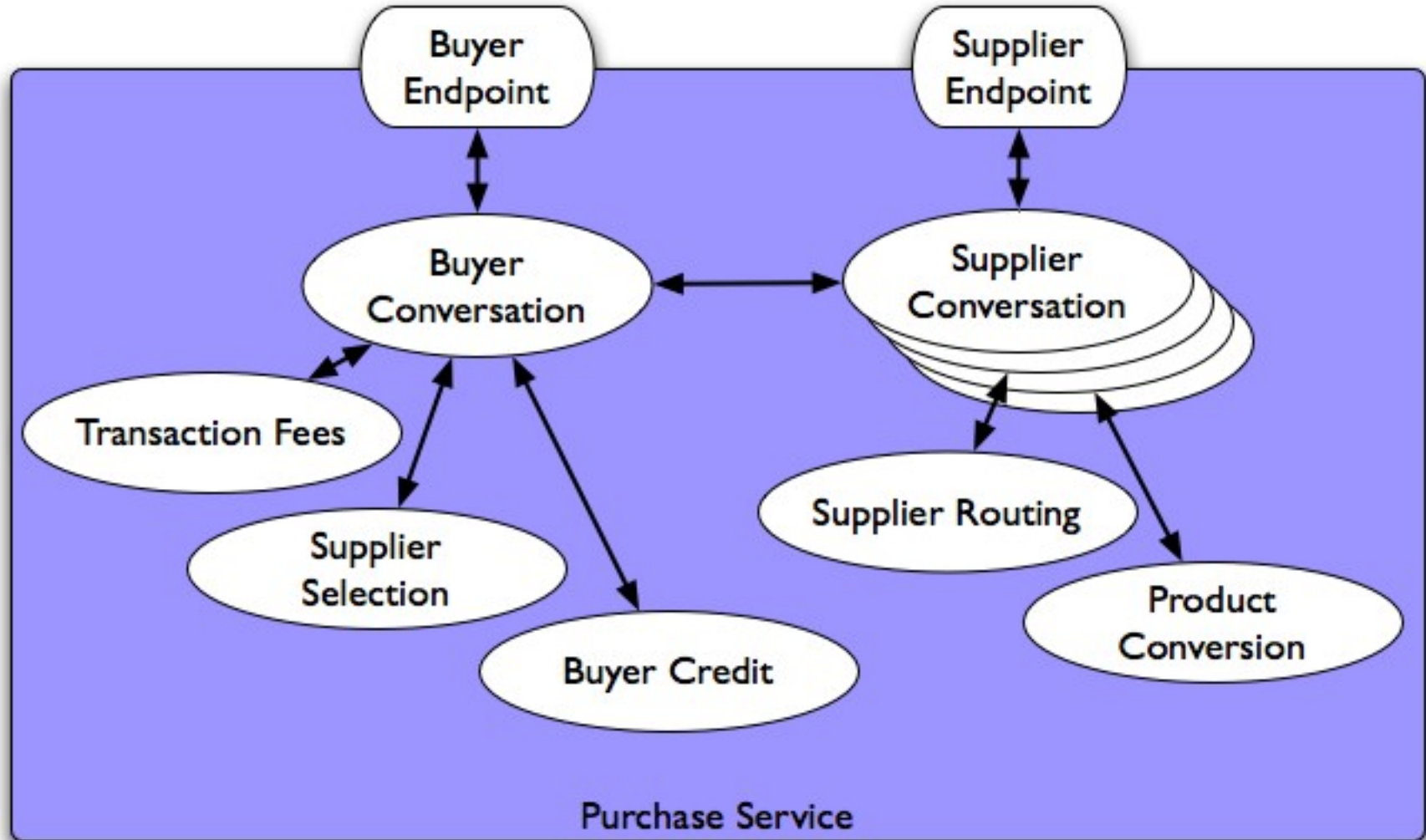
A Java EE Composite Application

Summary

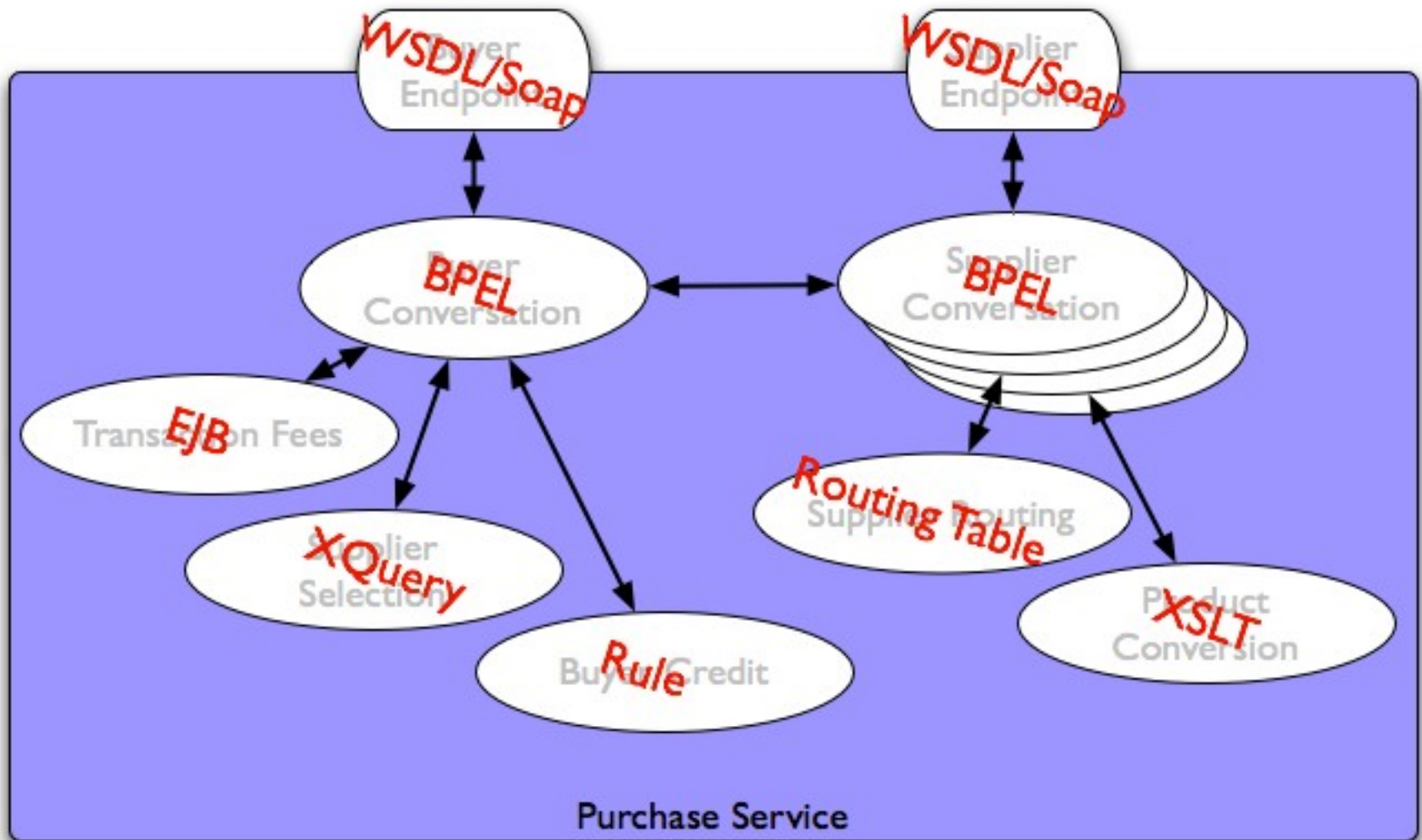
# Purchase Service



# Purchase Service Functions



# Purchase Service Implementation



# Composite Applications - summary

A composite application is a collection of existing and independently developed applications and new business logic, orchestrated together into a brand new solution of a business problem that none alone can solve.

Such an application looks to the user like a regular new interactive application, yet in reality it may be only 10 percent new and 90 percent an assembly of pre-existing components or data. The "glue" that brings a composite application together is integration technology.



# Agenda

Why Services

Why Composite Applications

**BPEL in the mix**

A Java EE Composite Application

Summary



# BPEL “Fixes” WSDL

WSDL: unordered set of operations

Operations are message exchanges

Need rules for ordering

Support for sequencing

Support for concurrency

Choreography with external entities

# BPEL is a Web Service Sequencing Language

Process defines “conversation” flow chart

Conversation consists of only WSDL-described message exchanges.

BPEL provides and consumes WSDL defined services

Process instance is a particular conversation following the chart

Execution systems can support multiple concurrent conversations.





# Agenda

Why Services

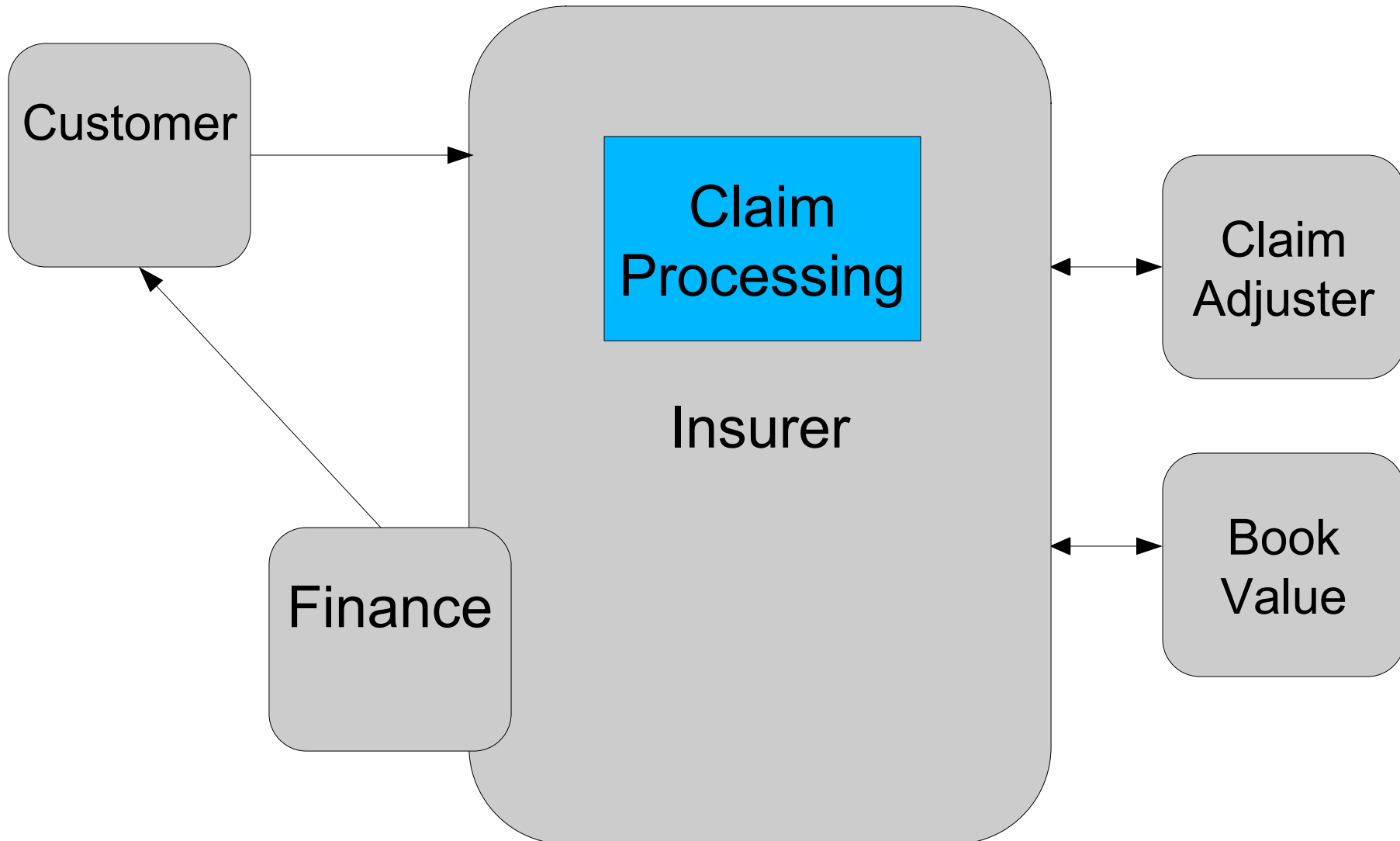
Why Composite Applications

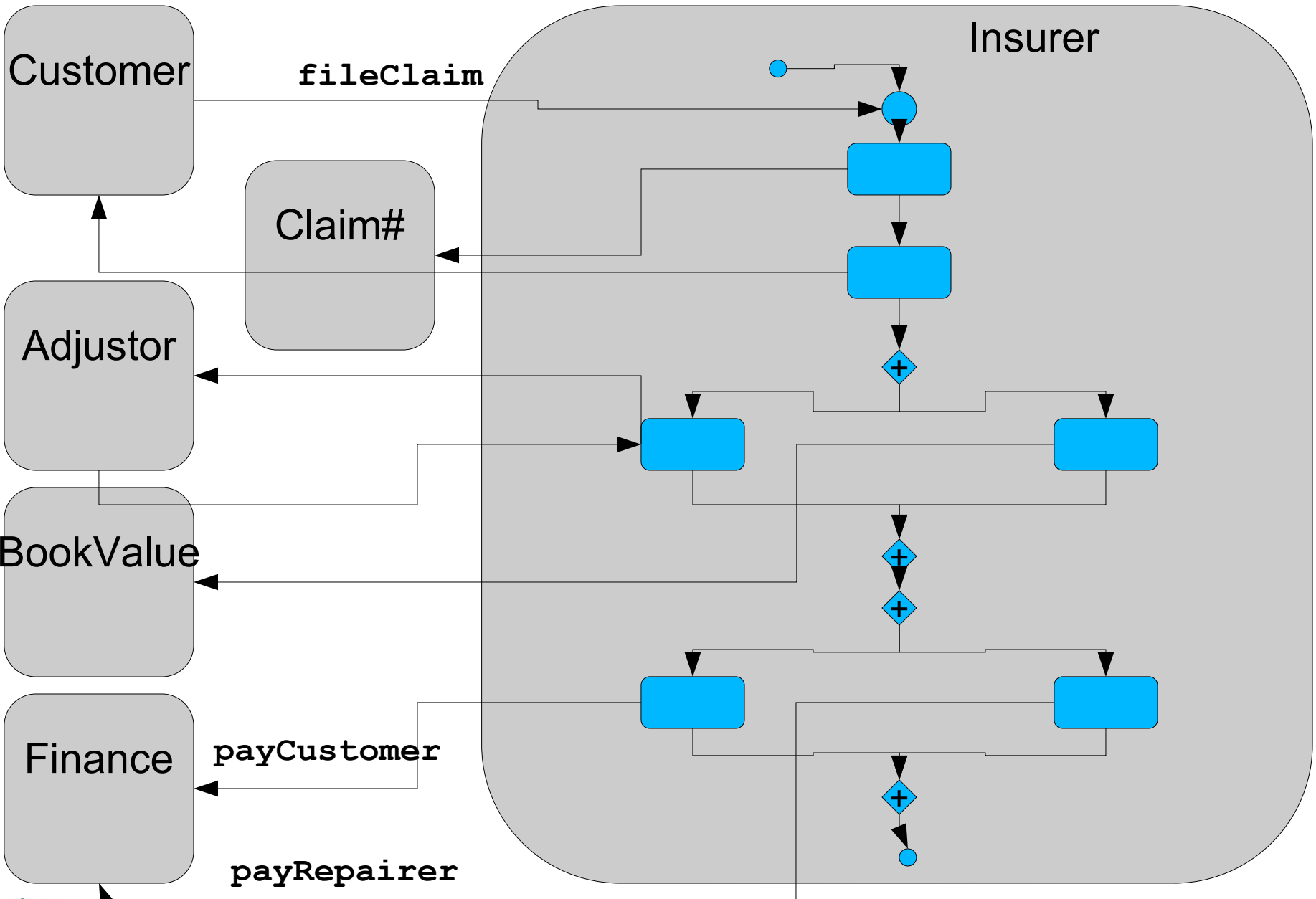
BPEL in the mix

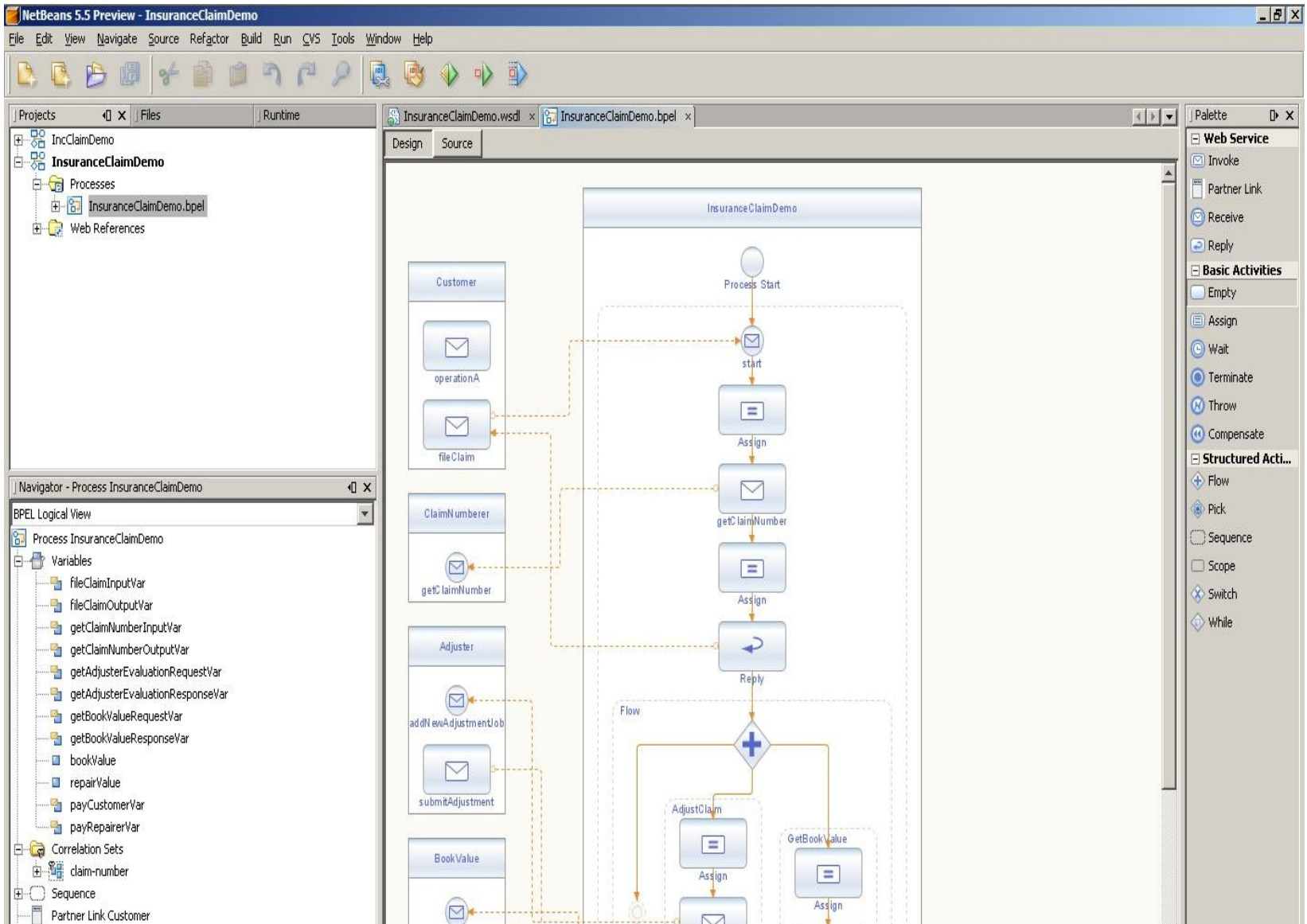
**A Java EE Composite Application**

Summary

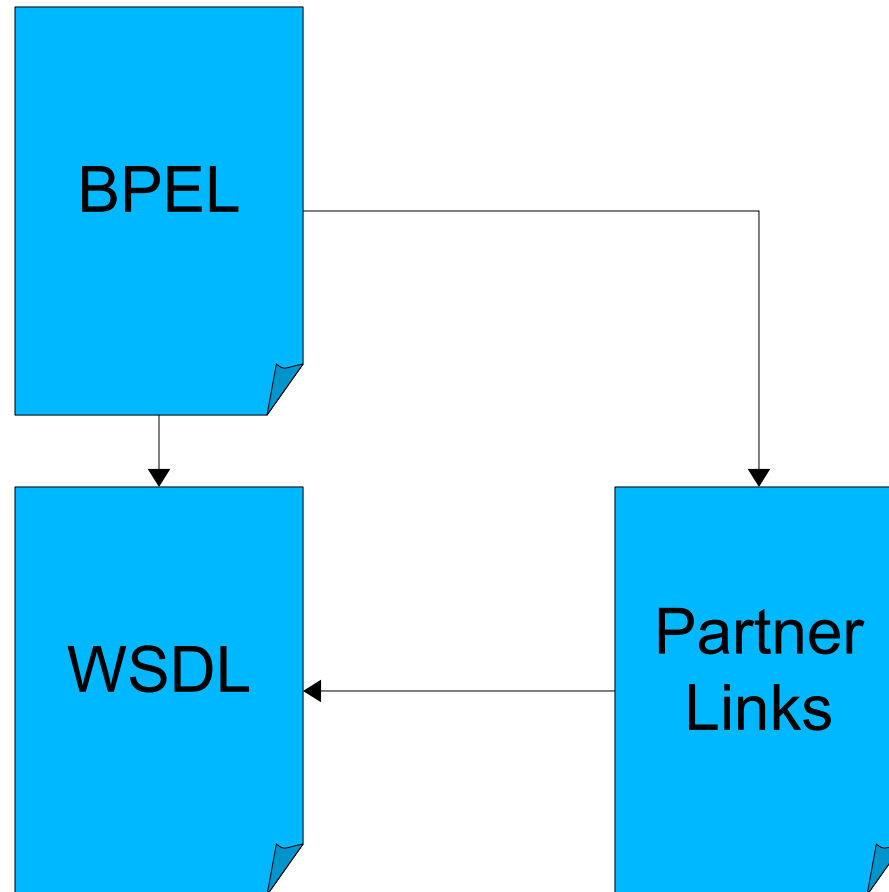
# Example Scenario







# What are the artifacts?



# WSDL: Schema Types (1)

```
<types>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
  targetNamespace="http://example.com/insurance-claim-1"
  xmlns:tns="http://example.com/insurance-claim-1">
    <xsd:element name="customer-id" type="xsd:string"/>
    <xsd:element name="car">
      <xsd:complexType>
        <xsd:attribute name="make" use="required"/>
        <xsd:attribute name="model" use="required"/>
        <xsd:attribute name="year" use="required"
          type="xsd:integer"/>
      </xsd:complexType>
    </xsd:element>
```

## WSDL: Schema Types (2)

```
<xsd:complexType name="tFileClaimRequest">
  <xsd:all>
    <xsd:element name="file-claim-request">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="tns:customer-id"/>
          <xsd:element ref="tns:car"/>
          <xsd:element ref="tns:description"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:all>
</xsd:complexType>
<xsd:element name="description" type="xsd:string"/>
```



## WSDL: Schema Types (3)

```
<xsd:complexType name="tFileClaimResponse">
  <xsd:all>
    <xsd:element name="file-claim-response">
      <xsd:complexType>
        <xsd:attribute name="claim-number"
          use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:all>
</xsd:complexType>
</xsd:schema>
</types>
```



# WSDL: Message definitions

```
<!--Message definitions used as parameters to operations  
in portType -->
```

```
<wsdl:message name="fileClaimRequestMessage">  
<wsdl:part name="request"  
element="xs:tFileClaimRequest" />  
</wsdl:message>
```

```
<wsdl:message name="fileClaimResponseMessage">  
<wsdl:part name="response" type="xs:tFileClaimResponse" /  
>  
</wsdl:message>
```



# WSDL: Port Types (implemented by BPEL process)

```
<wsdl:portType name="IcdClientPortType">  
  <wsdl:operation name="fileClaim">  
    <wsdl:input message="tns:fileClaimRequestMessage" />  
    <wsdl:output message="tns:fileClaimResponseMessage" />  
  </wsdl:operation>  
</wsdl:portType>
```



# WSDL: Partner Link Definitions

```
<plnk:partnerLinkType name="IcdClientPartnerLinkType" >  
  <plnk:role name="ClaimServiceProvider">  
    <plnk:portType name="tns:IcdClientPortType" />  
  </plnk:role>  
</plnk:partnerLinkType>
```



# BPEL: ...fileClaim

```
<sequence>
  <receive name="start" partnerLink="Customer"
    portType="wsdlNS:IcdClientPortType"
    operation="fileClaim" variable="fileClaimInputVar"
    createInstance="yes" />
  <assign>
    <copy>
      <from variable="fileClaimInputVar" />
      <to variable="getClaimNumberInputVar" />
    </copy>
  </assign>
```



## ...fileClaim continued

```
<invoke name="getClaimNumber" partnerLink="ClaimNumberer"
  portType="wsdlNS:ClaimNumbererPortType"
  operation="getClaimNumber"
  inputVariable="getClaimNumberInputVar"
  outputVariable="getClaimNumberOutputVar">
  <correlations>
    <correlation set="claim-number" initiate="yes" pattern="out"
  />
</correlations>
</invoke>
<assign>
  <copy>
    <from variable="getClaimNumberOutputVar" />
    <to variable="fileClaimOutputVar" />
  </copy>
</assign>
```



# BPEL: ...the reply

```
<reply partnerLink="Customer"  
portType="wsdl:INS:IcdClientPortType"  
operation="fileClaim" variable="fileClaimOutputVar" />  
  
<flow>  
  <sequence name="AdjustClaim">  
    ...  
  </sequence>  
  <sequence name="GetBookValue">  
    ...  
  </sequence>  
</flow>
```

# BPEL: ...the decision

```
<switch name="repair-it-or-write-it-off">
  <case condition="bpws:getVariableData('repairValue')
<= bpws:getVariableData('bookValue')">
    <sequence name="authorise-repair">
      ...
    </sequence>
  </case>
  <otherwise>
    <sequence name="write-off">
      ...
    </sequence>
  </otherwise>
</switch>
```

# Book Value Service (EJB 3.0)

```
@Stateless
@WebService(serviceName = "BookValueService",
targetNamespace="http://www.mycomp.org/InsuranceClaimDemo")
public class BookValueService {
    /**
     * Returns the book value for the given automobile type
     * @param make manufacturer of the given automobile
     * @param model model of the given automobile
     * @param year year of the given automobile
     * @return book value of the given automobile type
     */
    @WebMethod(operationName = "getBookValue")
    float getBookValue(String make, String model, int year)
    {
        ...
    }
}
```





# Service Consumption (JAX-WS 2.0)

1. Generate Java binding to web service exposed to the customer:

```
wsimport -d ../gen -p org.mycomp.ins.claim.demo  
InsuranceClaimDemo.wsdl
```

2. Use the binding in application code:

```
/** JAXWS-generated service for accessing insurance claim svc */  
private final IcdClientPortType insClaimSvc_  
  
public String submitClaim(String customerId, String make,  
    String model, int year) {  
    TFileClaimRequest req =  
        ObjectFactory.createTFileClaimRequest();  
    // ... fill in request with make,model,year,customerId.  
  
    TFileClaimResponse resp = insClaimSvc_.fileClaim(req);  
  
return resp.getClaimNumber();  
}
```

# Runtime: Java™ EE and JBI

JBI serves as messaging infrastructure:

- Java EE web services interact through JBI

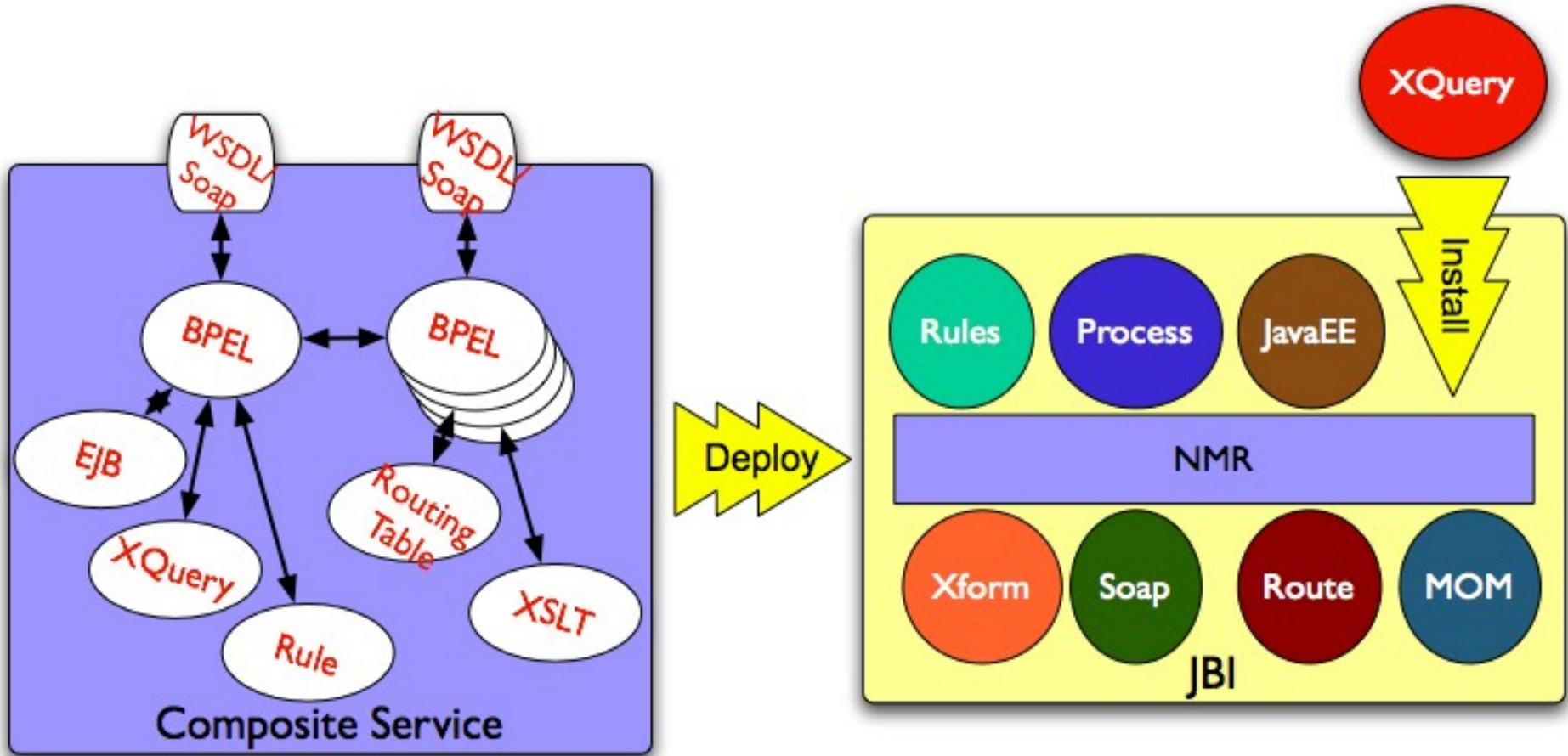
- JBI bindings allow remote consumers and providers

- Add other service technologies as JBI components

Transparent to Java EE programmer

- Reuse without re-coding

# Java™ Business Integration (JSR 208)



# DEMO

Add title here

*Instructions:  
(Delete this red box before  
submitting your slides)*

*Use this slide to introduce  
a demo in your presentation.  
Simply duplicate this slide to  
create additional demo slides.*



# Agenda

Why Services

Why Composite Applications

BPEL in the mix

A Java EE Composite Application

Summary

# Summary

Services can be created and used using a variety of Java EE technologies

BPEL is a service orchestration language for creating stateful composite applications

Services can be reused

Services can be re-implemented using other technologies as long as service interface is preserved without changing consumers

JBI is the enabling infrastructure

# For More Information

## List

### Other Technical Sessions

TS-1076 “Practical SOA Business Integration Using OpenESB:  
A Distributed Java Business Integration Composed Services  
Application How-to and Demo”

Panel TS-2002 “What Is Happening With SOA in Open Source?”

### BOFs

BOF-0089 “What’s Next for Java Business Integration (JBI)?”

### URLs

[java.sun.com/integration](http://java.sun.com/integration)

<http://www.sun.com/products/soa>

<http://developers.sun.com/prodtech/javatools/jsenterprise>

# Q&A

<code/>





the  
**POWER**  
of  
**JAVA™**



JavaOne  
Part of the Oracle and Sun Microsystems

# Building a service with BPEL and Java EE: How Composite Apps and JBI simplify SOA development

Ron Ten-Hove , Peter Walker and Gopalan Raj

Senior Staff Engineers

Sun Microsystems, Inc. [java.sun.com/integration](http://java.sun.com/integration)

TS-3175