Building a Service with BPEL and Java™ EE: How Composite Apps and Java Business Integration Simplify SOA Development

Ron Ten-Hove, Peter Walker, Gopalan Raj

Senior Staff Engineers
Sun Microsystems, Inc.
java.sun.com/integration

Session TS-3175

Goal

This session describes how developers using Java™ Platform, Enterprise Edition can create composite applications using BPEL with current Java EE tools and application servers that support Java Business Integration

Agenda

Why Services
Why Composite Applications
BPEL in the Mix
A Java EE Based Composite Application
Summary

Agenda

Why Services

Why Composite Applications

BPEL in the Mix

A Java EE Based Composite Application

Summary

Why Services?

- SOA = an architectural principle for structuring systems that
- 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
- Enterprise JavaBeans™ (EJB™) technology
- BPEL
- SQL
- XQuery
- Routing table
- Business rules
- EDI transform
- …

Agenda

Why Services

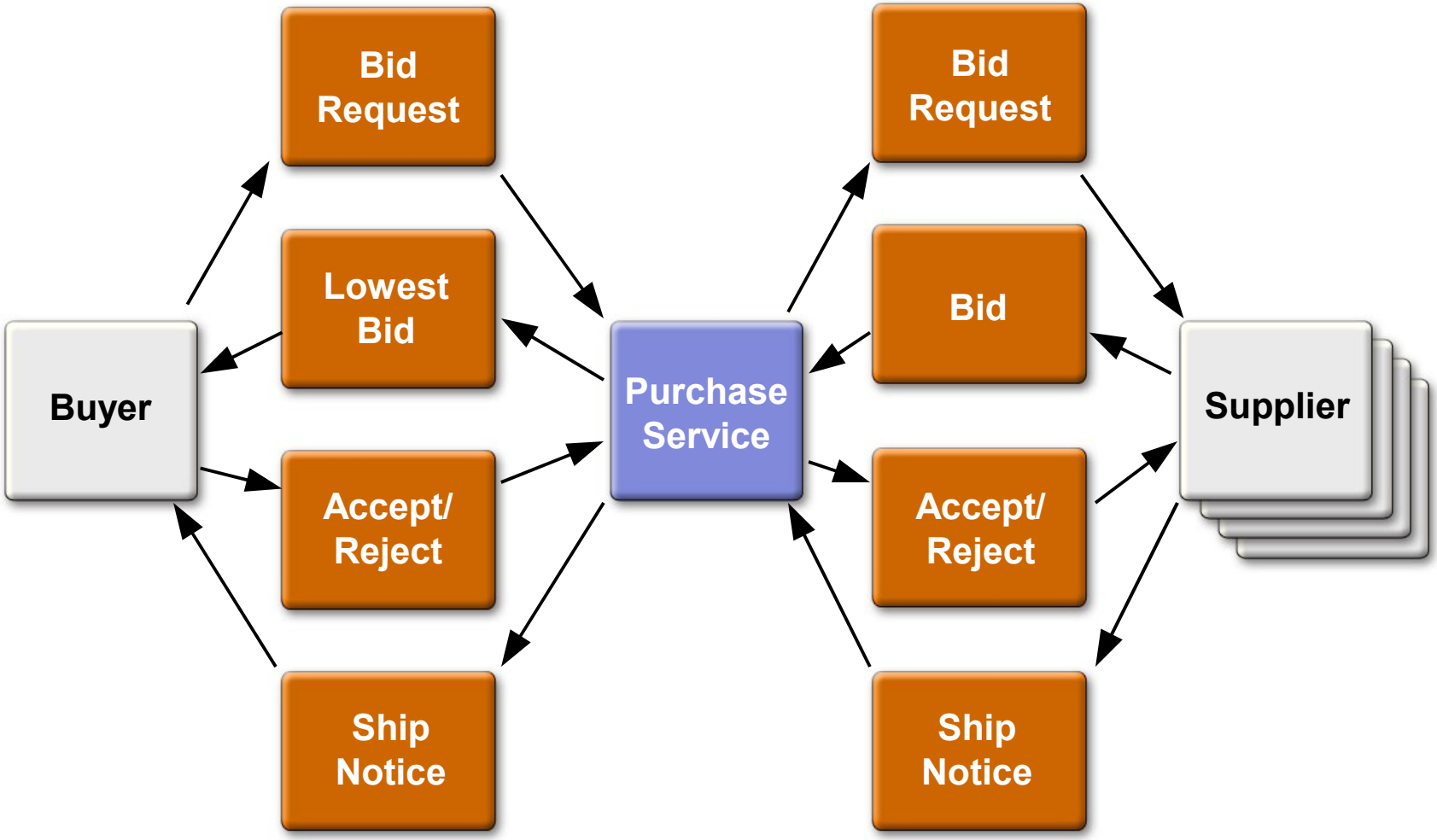
Why Composite Applications

BPEL in the Mix

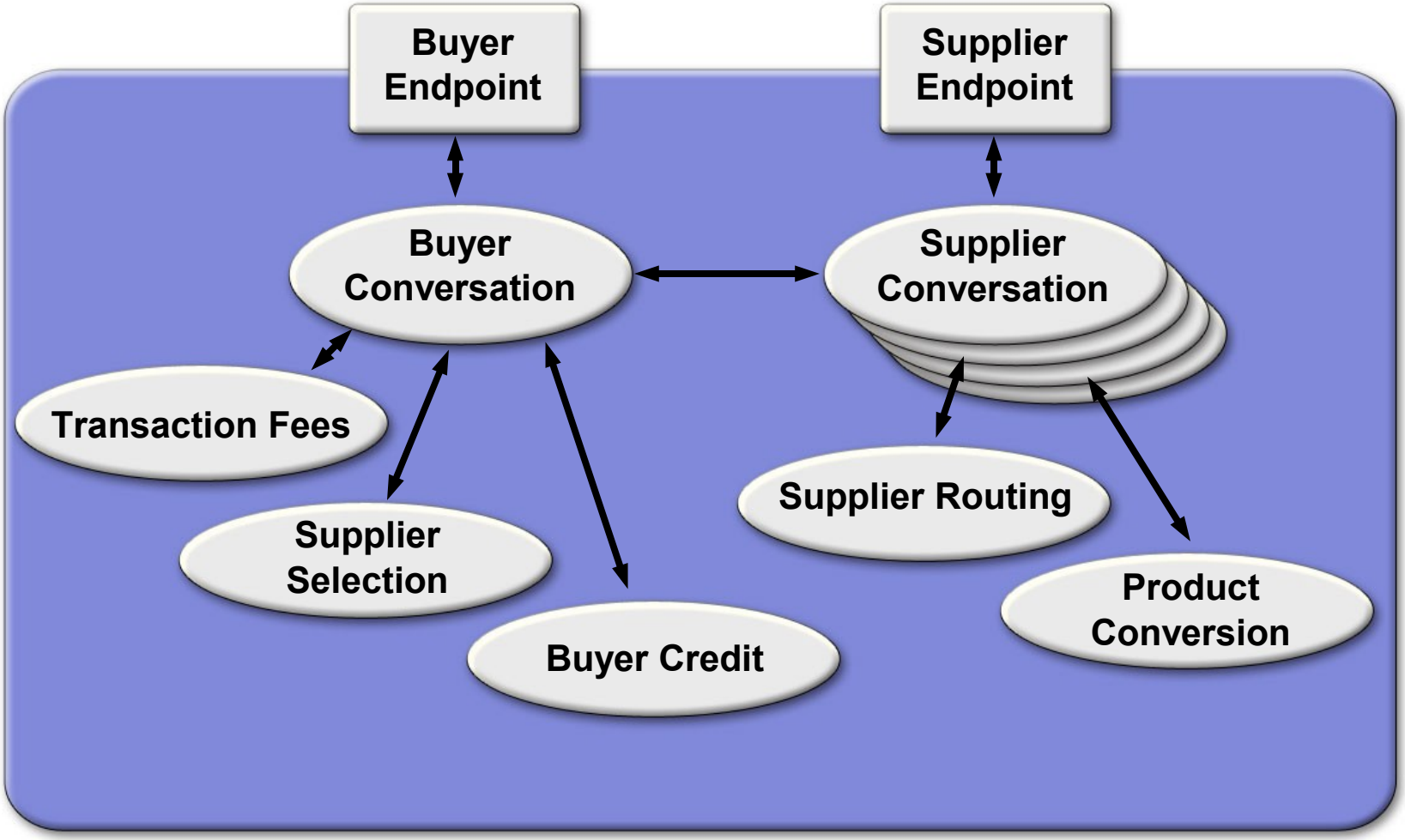
A Java EE Based Composite Application

Summary

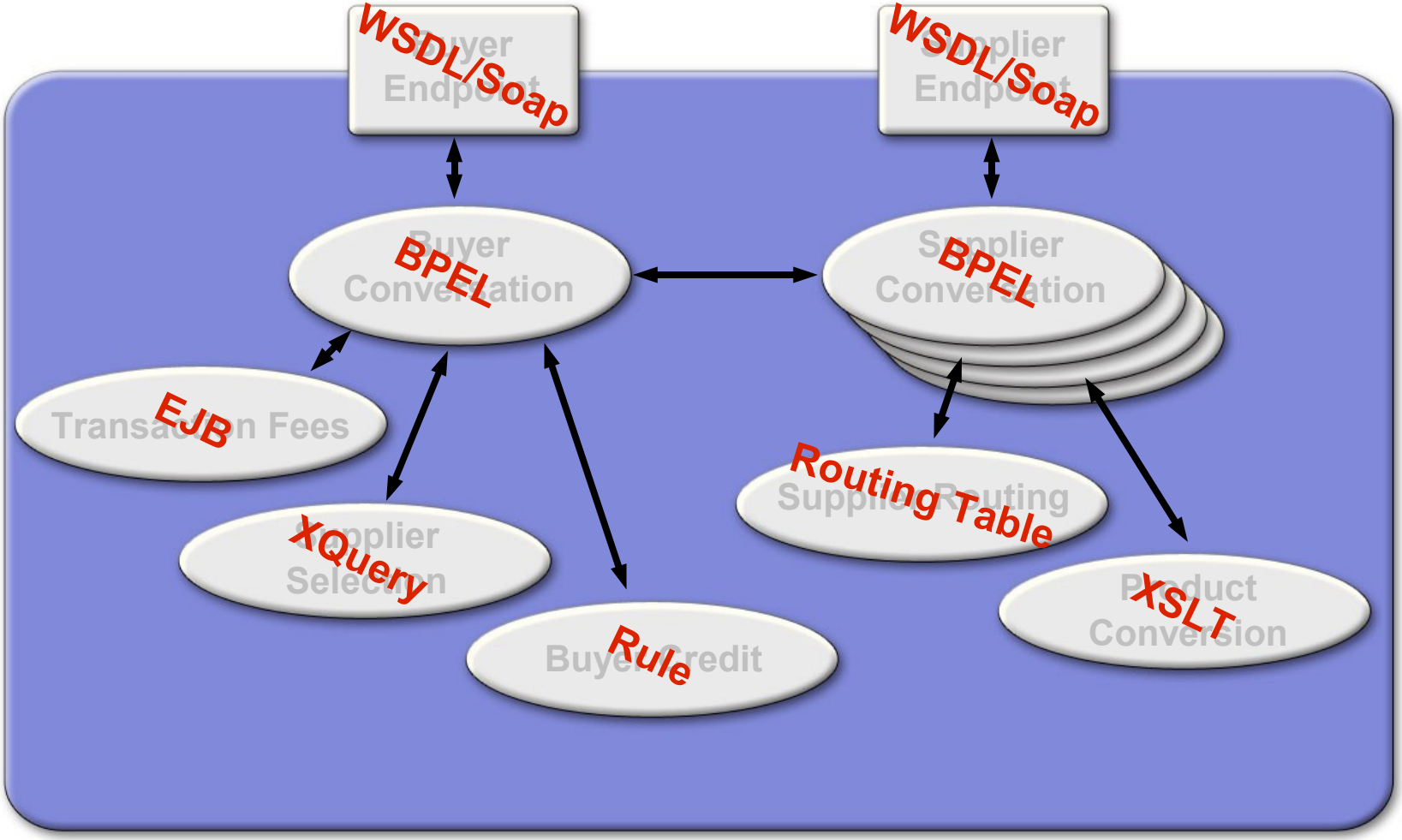
Purchase Service



Purchase Service Functions



Purchase Service Functions



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 Based 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

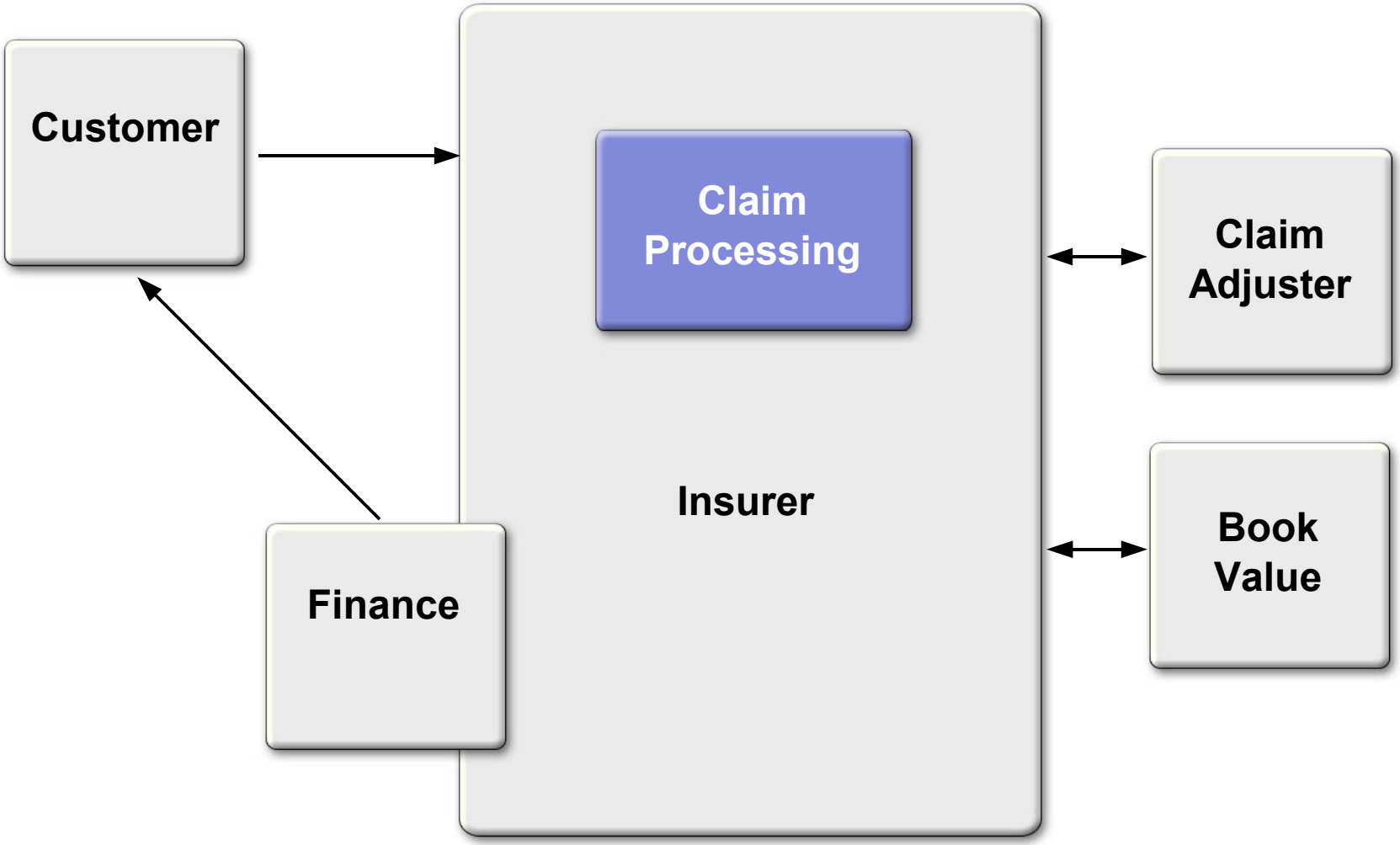
Why Composite Applications

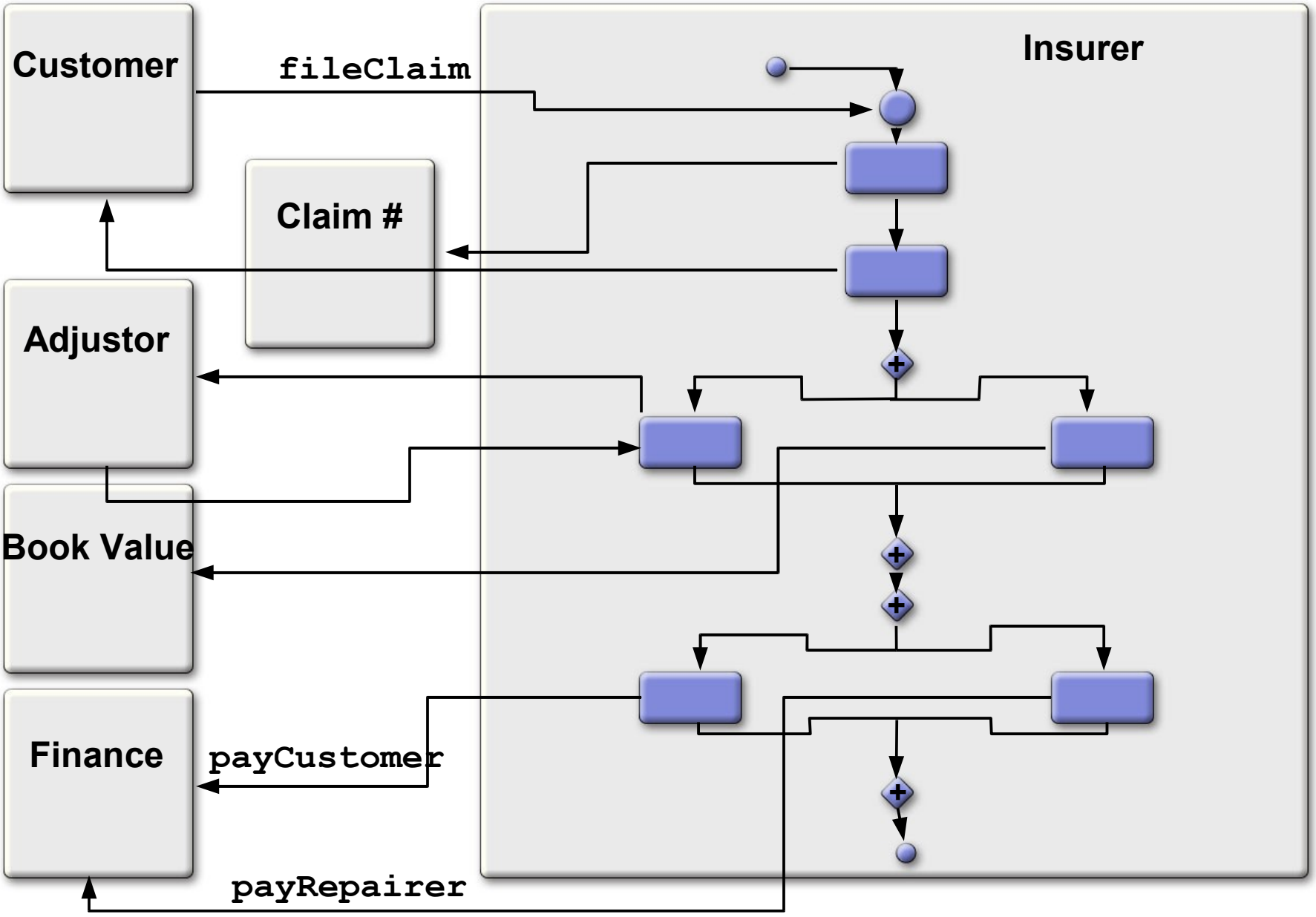
BPEL in the Mix

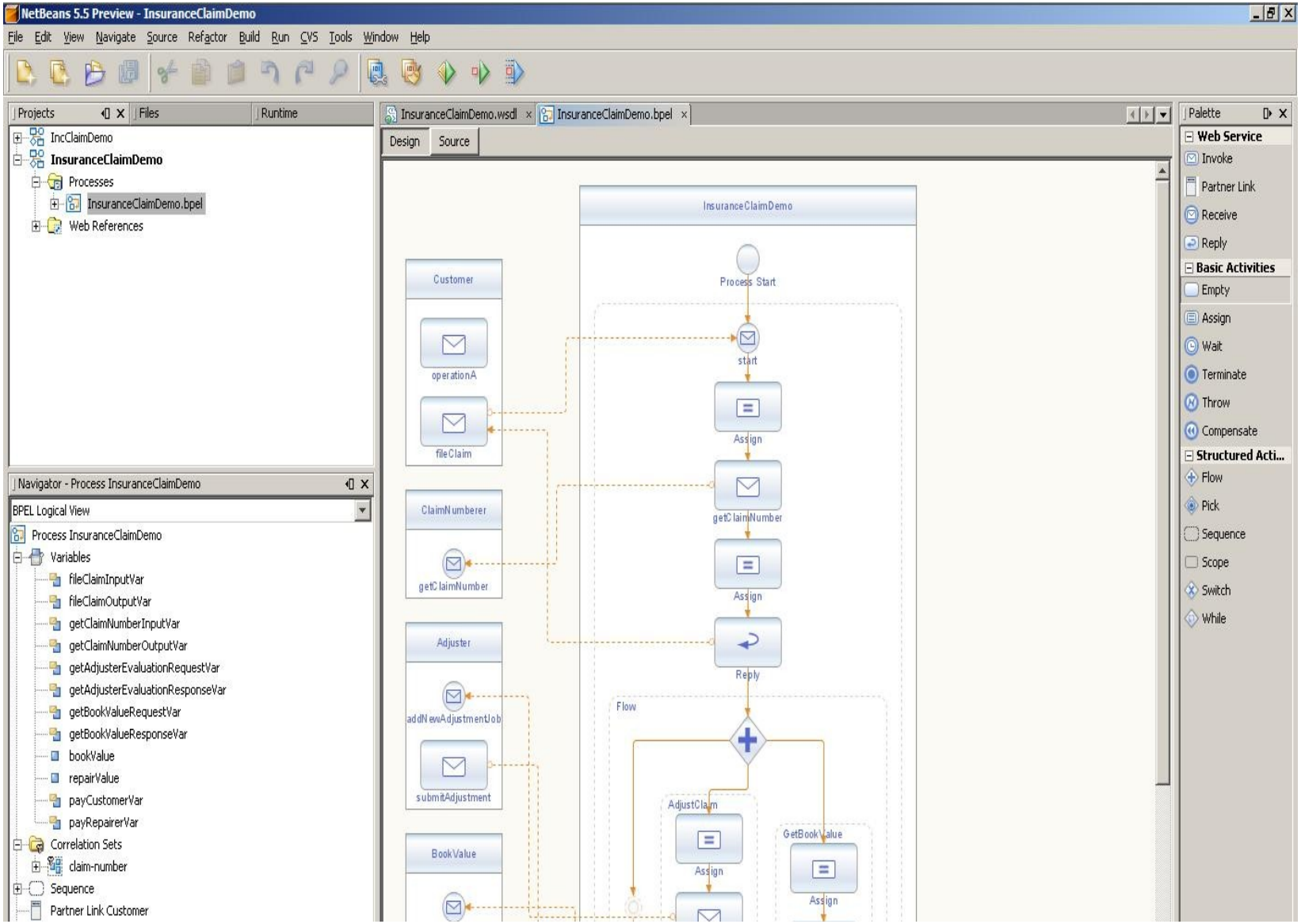
A Java EE Based Composite Application

Summary

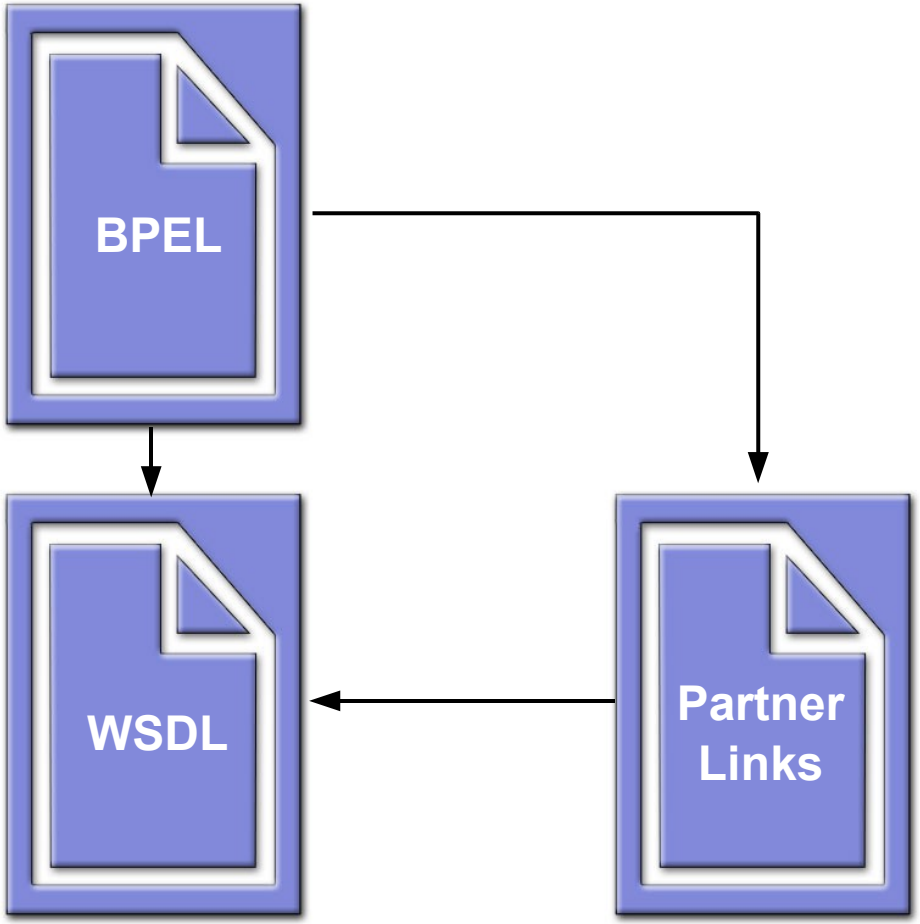
Example Scenario




Customer
fileClaim
Insurer
Claim #
Adjustor
Book Value
Finance
payCustomer
payRepairer


NetBeans 5.5 Preview - InsuranceClaimDemo
File Edit View Navigate Source Refactor Build Run CVS Tools Window Help
Projects
Files
Runtime
IncClaimDemo
InsuranceClaimDemo
Processes
InsuranceClaimDemo.bpel
Web References
Navigator - Process InsuranceClaimDemo
BPEL Logical View
Process InsuranceClaimDemo
Variables
fileClaimInputVar
fileClaimOutputVar
getClaimNumberInputVar
getClaimNumberOutputVar
getAdjusterEvaluationRequestVar
getAdjusterEvaluationResponseVar
getBookValueRequestVar
getBookValueResponseVar
bookValue
repairValue
payCustomerVar
payRepairerVar
Correlation Sets
claim-number
Sequence
Partner Link Customer
InsuranceClaimDemo.wsdl
InsuranceClaimDemo.bpel
Design
Source
InsuranceClaimDemo
Customer
operationA
fileClaim
ClaimNumberer
getClaimNumber
Adjuster
addNewAdjustmentJob
submitAdjustment
BookValue
Process Start
start
Assign
getClaimNumber
Assign
Reply
Flow
AdjustClaim
Assign
GetBookValue
Assign
Palette
Web Service
Invoke
Partner Link
Receive
Reply
Basic Activities
Empty
Assign
Wait
Terminate
Throw
Compensate
Structured Acti...
Flow
Pick
Sequence
Scope
Switch
While

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:partnerLinkTypename="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 (Cont.)

```
<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="wsdlNS: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 Technology)

```
@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 (Java APIs) for XML Web Services (JAX-WS 2.0)

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

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

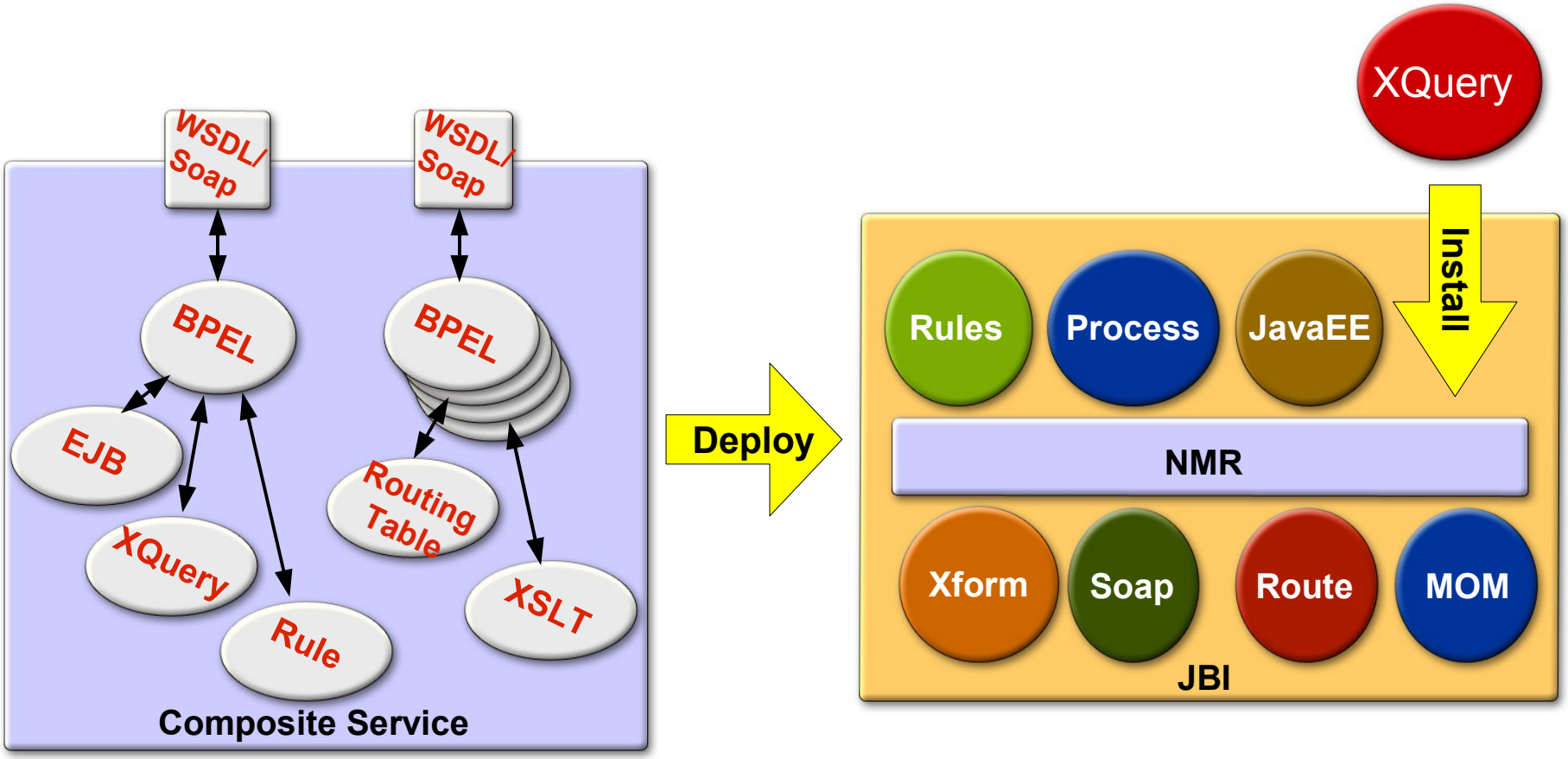
2. Use the binding in application code:

```
/** 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 Platform and Java Business Integration

- Java Business Integration serves as messaging infrastructure
  - Java EE web services interact through Java Business Integration
  - Java Business Integration bindings allow remote consumers and providers
  - Add other service technologies as Java Business Integration components
- Transparent to programmer using Java EE technology
  - Reuse without re-coding

Java Business Integration (JSR 208)



Agenda

Why Services

Why Composite Applications

BPEL in the Mix

A Java EE Based 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
- Java Business Integration 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://www.sun.com/products/soa
  - http://developers.sun.com/prodtech/javatools/jsenterprise

Q&A

Building a Service with BPEL and Java™ EE: How Composite Apps and Java Business Integration Simplify SOA Development

Ron Ten-Hove, Peter Walker, Gopalan Raj

Senior Staff Engineers
Sun Microsystems, Inc.
java.sun.com/integration

Session TS-3175