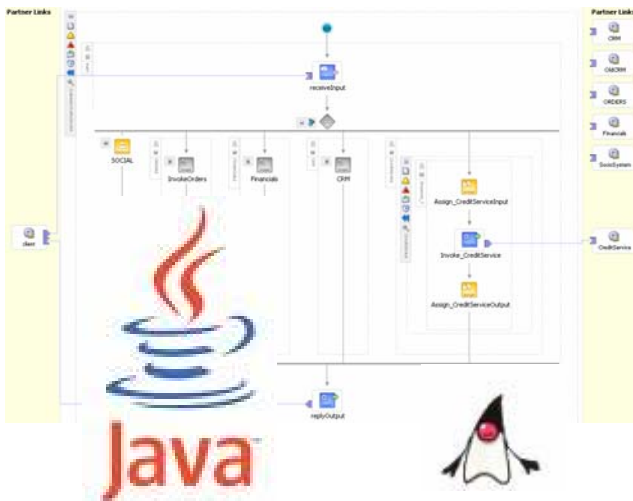


How Java and BPEL join forces

What every Java developer should know about BPEL

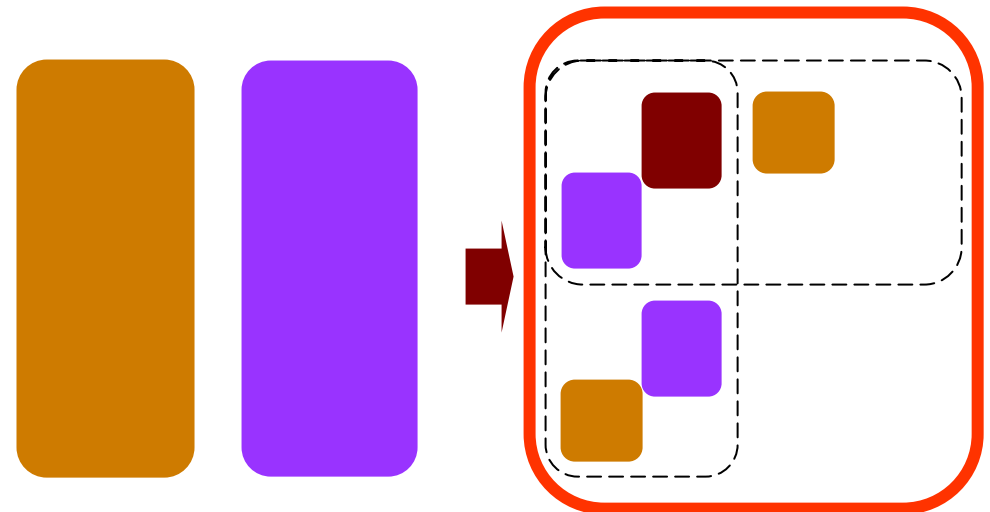


Lucas Jellema (AMIS)
NL-JUG's J-Spring 2006
Thursday 15th june

- Introduction of Service Oriented Architecture and BPEL (Business Process Execution Language)
- *Calling BPEL Services from Java Applications*
 - When, Why and How?
- *Invoking Java based Services from a BPEL Process*
 - When, Why and How?
- Workflow Tasks picked up and performed by Java applications
 - Human Workflow and Complex Automated Workflow
- Conclusions

- Attitude, Way of thinking
 - No individual, stand-alone applications
 - Instead:
 - Implementations of business processes
 - composition of services
 - Optional: execution in a container

- Reuse of services
 - Loosely coupled
 - Standardized interfaces
 - Central management facilities



- Library (dll, jar)
- EJB
- URL
- RSS feed
- Portlet
- SOAP WebService
- Database View
-



If it exposes a clear, standardized *interface*





*Are we back in 2000 => 1995 => 19990 =>?
 What is new? Why is this such a hype?*

● Standards!

- Industry-wide, supported by major vendors and governing bodies

- XML
- Interfaces & protocols
 - From CORBA, DCOM, RMI to JMS, JCA and most importantly SOAP Web Services
 - Internet & HTTP
- Cross Technology Bridges
 - Apache WSIF, SOAP WebServices, Adapters in MOM, ESB and BPEL
- Formal service specification using WSDL
- Tool and Programming-language support
 - IDEs, Frameworks, Libraries, Browsers
- Infrastructures
 - BPEL Container, Enterprise Service Bus, Web Service Management



● Money

- Lower IT development costs due to high level of reuse
- Cost reduction through less manual action in business process execution



● Time

- Shorter time to market of business processes
- Quicker execution of business processes

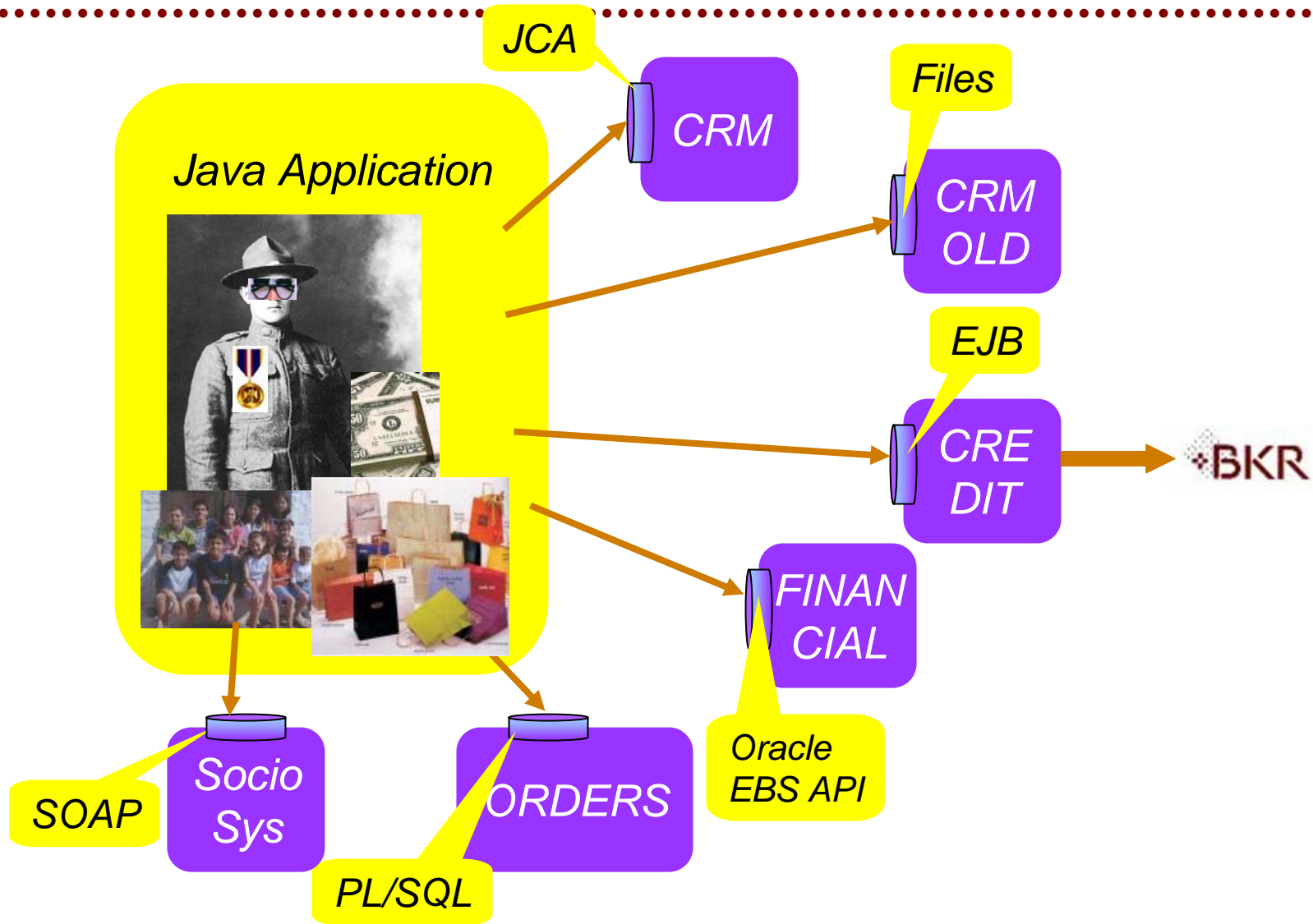


● Quality

- Higher quality systems through reuse of proven components and services
- Better process execution because of better interfaces and reduced human intervention

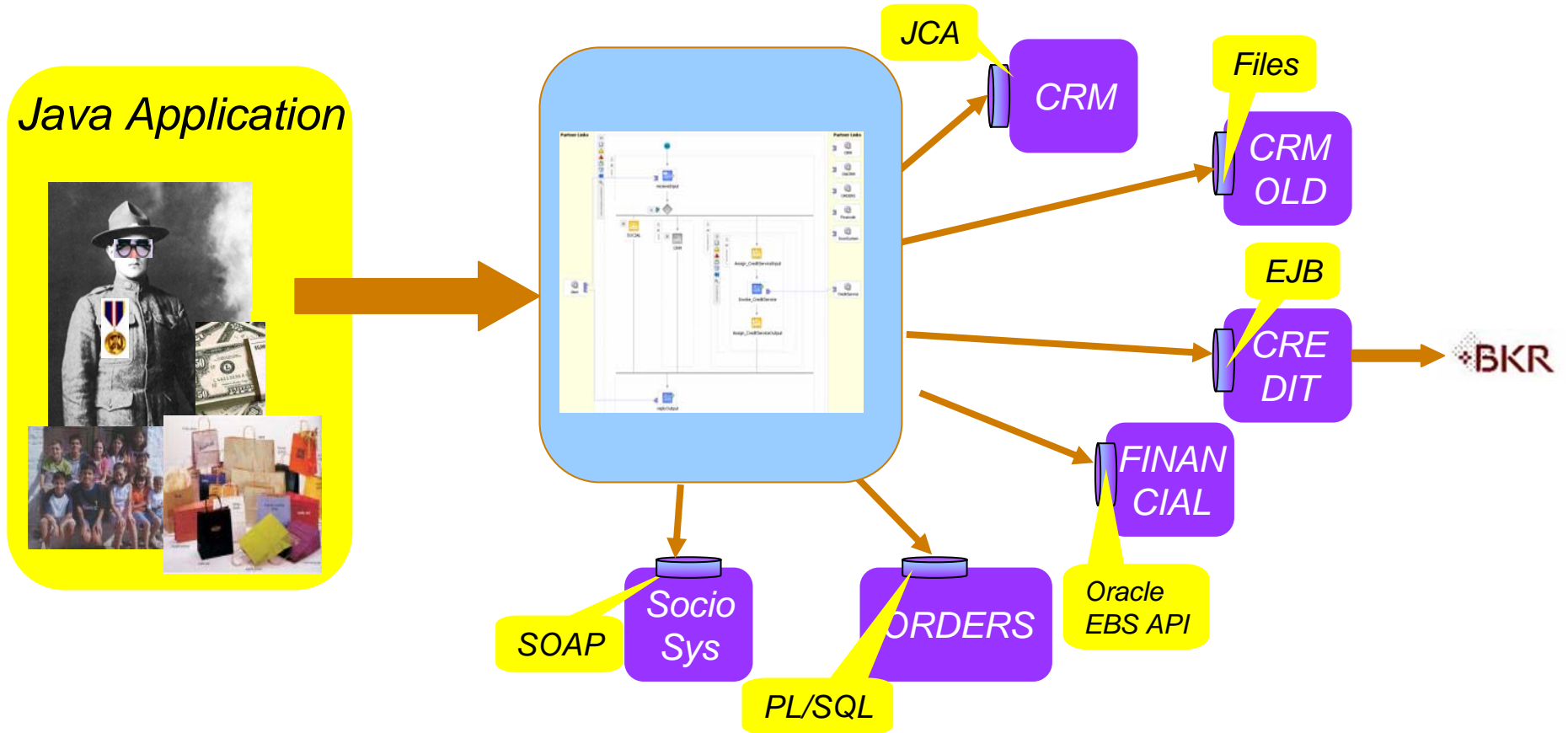
- Introduction of Service Oriented Architecture and BPEL (Business Process Execution Language)
- **Calling BPEL Services from Java Applications**
 - When, Why and How?
- Invoking Java based Services from a BPEL Process
 - When, Why and How?
- Workflow Tasks picked up and performed by Java applications
 - Human Workflow and Complex Automated Workflow
- Conclusions

Java application is looking for a 360 degrees customer view



- The number of technologies to interface with
 - Skills required
 - Complexity of resulting code
- The number of data structures and formats
- The very real danger of changes
 - Moving service endpoints
 - Changing Service APIs
- Reusing the “360 degrees Customer profile” service
- Monitor (trace, audit, debug) service execution

SOA-style approach: Create a BPEL Service as intermediary



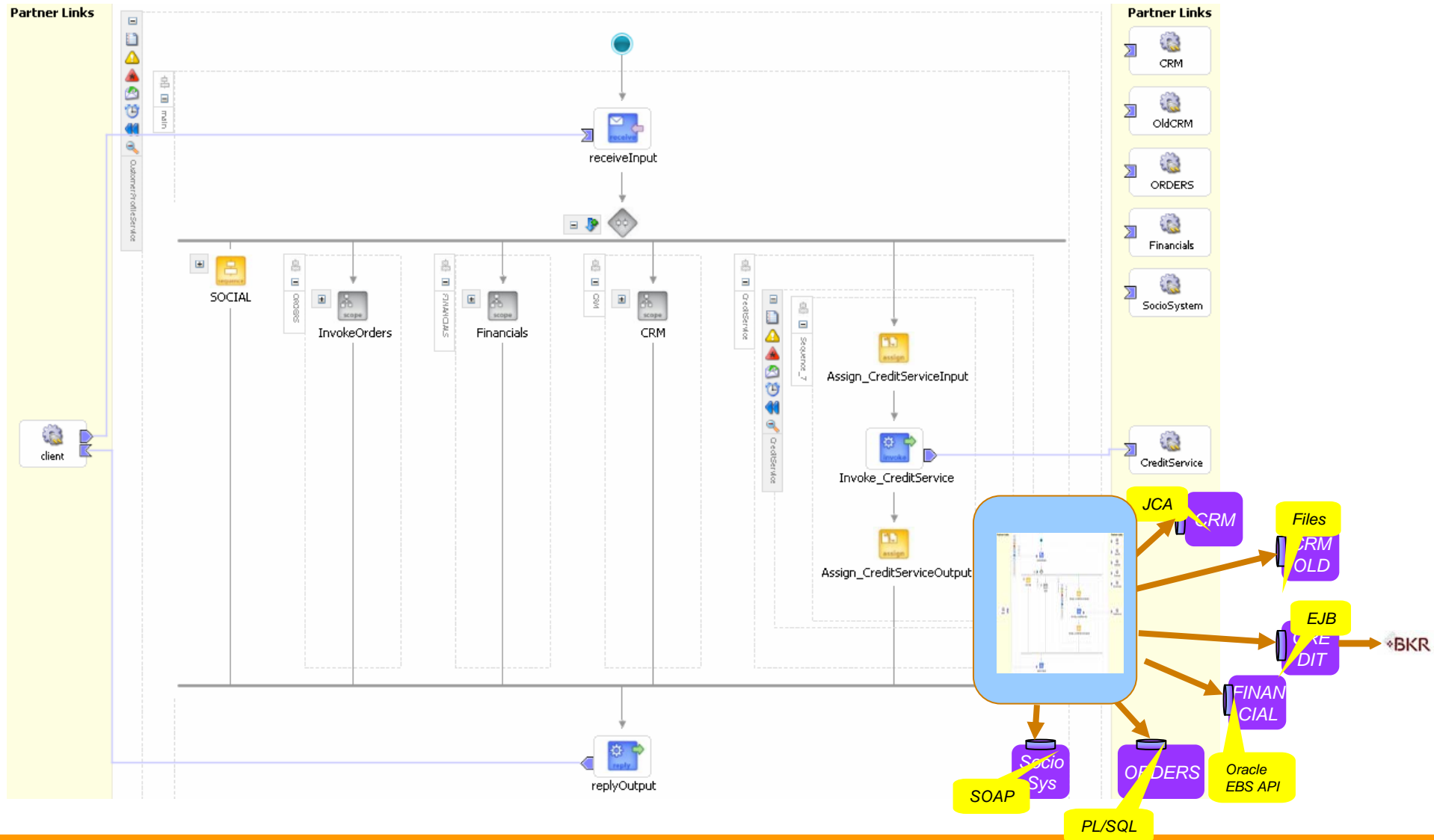
- Note: could also be done using an Enterprise Service Bus!

- Calling a SOAP WebService
 - For example using Apache AXIS, JAX-RPC
- Calling a remote Enterprise Java Bean (Session Bean) from another JVM
 - Generic Java API
- Making a local (intra JVM) call using the local interface
 - Generic Java API

decoupling

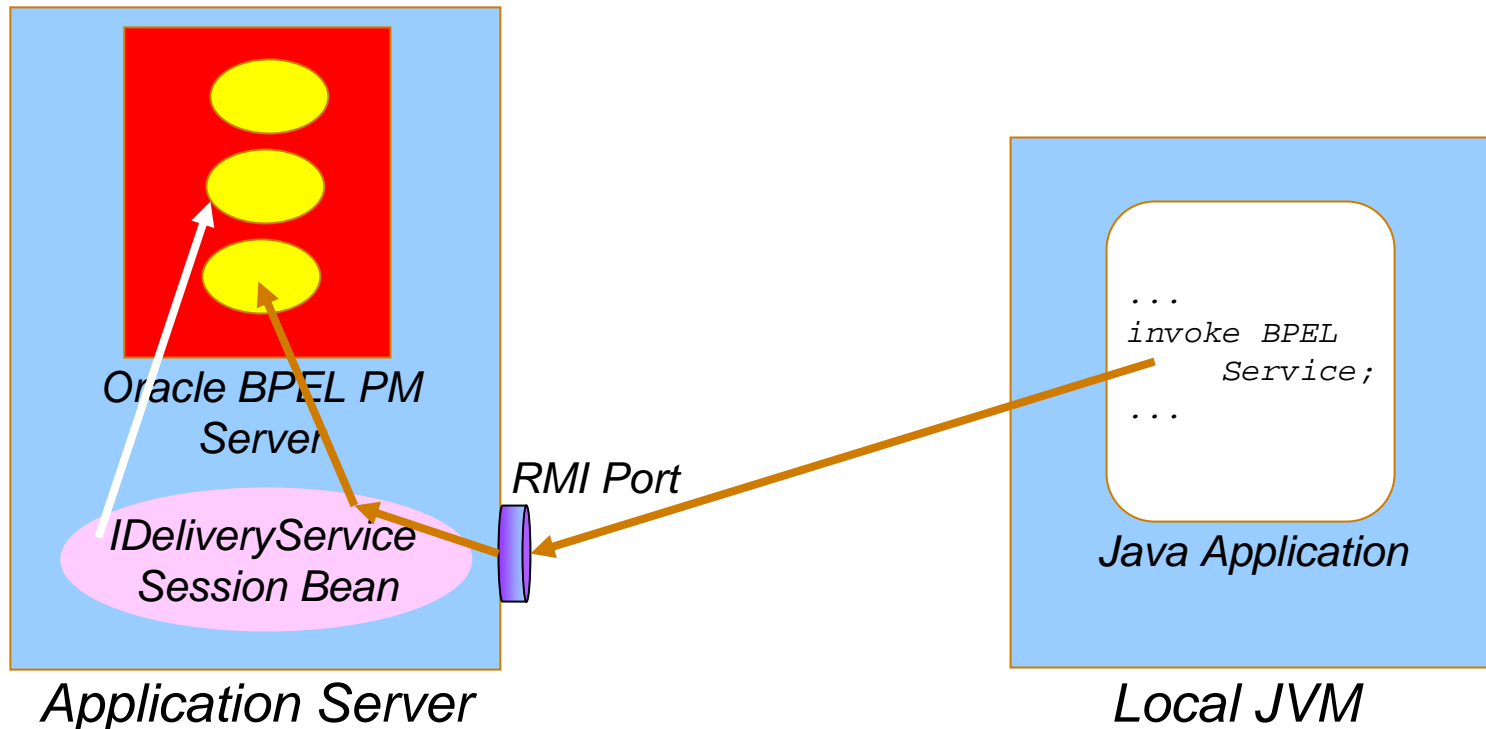


performance



Calling BPEL Service from a Java Application using RMI and BPEL Session Bean

- Java Application is local client
 - That invokes a (remote) service
 - possibly In a different JVM



```
// Set JNDI Properties for remote invocation ( will work for local also...
Hashtable jndi = null;
jndi = new Hashtable();
// Change the PROVIDER_URL to your BPEL PM host...
jndi.put(Context.PROVIDER_URL, "ormi://localhost/orabpel");
jndi.put(Context.INITIAL_CONTEXT_FACTORY, "com.evermind.server.rmi.RMIInitialContextFactory");
// connect to the admin principal
jndi.put(Context.SECURITY_PRINCIPAL, "admin");
jndi.put(Context.SECURITY_CREDENTIALS, "welcome");
jndi.put("dedicated.connection", "true");
// call to default domain using (default) password bpel
// note: if we run this code in the same JVM as the Oracle BPEL PM, then jndi can be null
Locator locator = new Locator("default", "bpel", jndi);
IDeliveryService deliveryService =
    (IDeliveryService) locator.lookupService(IDeliveryService.SERVICE_NAME);
```

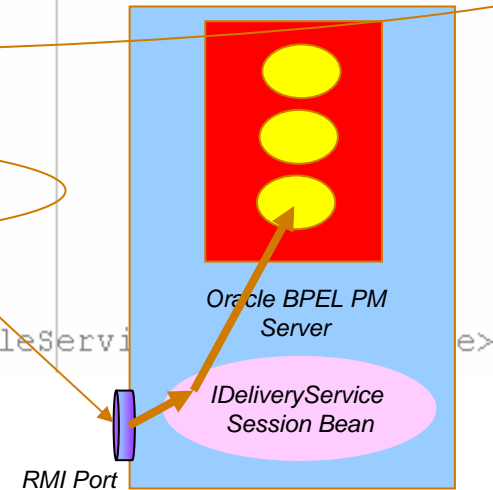
```

// construct the normalized message and send to collaxa server
NormalizedMessage nm = new NormalizedMessage();
String xml =
    "<CustomerProfileServiceProcessRequest xmlns=\"http://xmlns.oracle.com/CustomerProfileService\"
    customerId +
    "><customerId>" +
    "</customerId></CustomerProfileServiceProcessRequest> ";
nm.addPart("payload", xml);

NormalizedMessage res =
    deliveryService.request("CustomerProfileService", "process", nm);

Map payload = res.getPayload();
Element partEl = (Element)payload.get("payload"); //== <CustomerProfileService
//== <CustomerProfileServiceProcessResponse>
    <CustomerProfileServiceProcessResponse>
        <contactDetails>
            <name>Henk Willemsen</name>
            <street>Roodbruin</street>
        </contactDetails>
        <financialStatus/>

```



- Create a Java Client GUI or a Java Web Application that exposes the BPEL Service
- Using Java Server Faces – ADF Faces in Oracle JDeveloper 10.1.3
 - Create BPEL WebService DataControl from WSDL
 - Create new JSF JSP page
 - Drag & Drop UI Components based on DataControl
 - Run

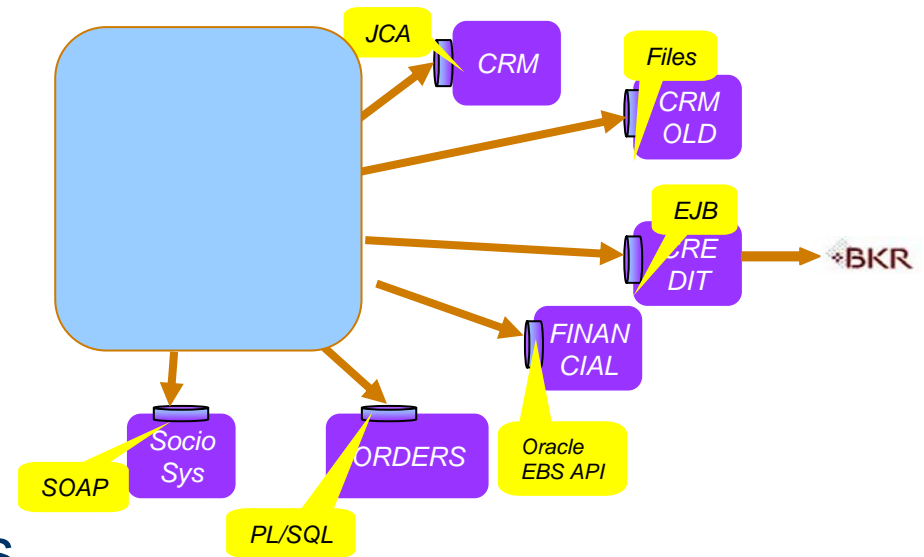
process_customerId	<input type="text" value="2"/>
<input type="button" value="process"/>	
name	Henk Willemsen
street	Roodbruin
houseNumber	24
postalCode	2716NK
city	Zoetermeer
country	nl
birthdate	
emailAddress	henkie@dotnot.com
telephone	0031-79943111
wantsMailings	N
customerStatus	BRONZE

- Introduction of Service Oriented Architecture and BPEL (Business Process Execution Language)
- Calling BPEL Services from Java Applications
 - When, Why and How?
- **Invoking Java based Services from a BPEL Process**
 - When, Why and How?
- Human Workflow Tasks
 - Putting a (Java Server) Face on Web Services and SOA
- Conclusions

- A BPEL Process is typically a combination of
 - XML data manipulation
 - Simple flow logic Loop, Decision Point, Wait,...
 - Service Calls

- Services can be
 - implemented in virtually any technology
 - invoked through a substantial number of protocols and adapters

- HTTP, EJB, JCA, JMS, Oracle Apps, Email, FTP, File, Fax, SMS, Java (WSIF), ...



- We will see a shift from ‘developing applications’ to ‘implementing business processes’
- Business processes are implemented as a chain of service calls with some orchestration on top
- Services do most of the actual work
 - Providing and processing data from enterprise stores
 - Calculations, Validations and Conversions
 - Reporting and Interacting
- A large portion of the functionality required by business processes is already available
 - In existing (legacy) applications
 - In a plethora of technologies, including Java
- Exposing existing functionality as reusable service is key

- Reusable functionality implemented in Java can have one or more of several interface styles
 - public method on POJO
 - Servlet
 - EJB
 - SOAP WS
- A BPEL Process can invoke each of these interface styles
- A BPEL Process can act as wrapper for Java based Services
 - Service (URL and Protocol) Virtualization

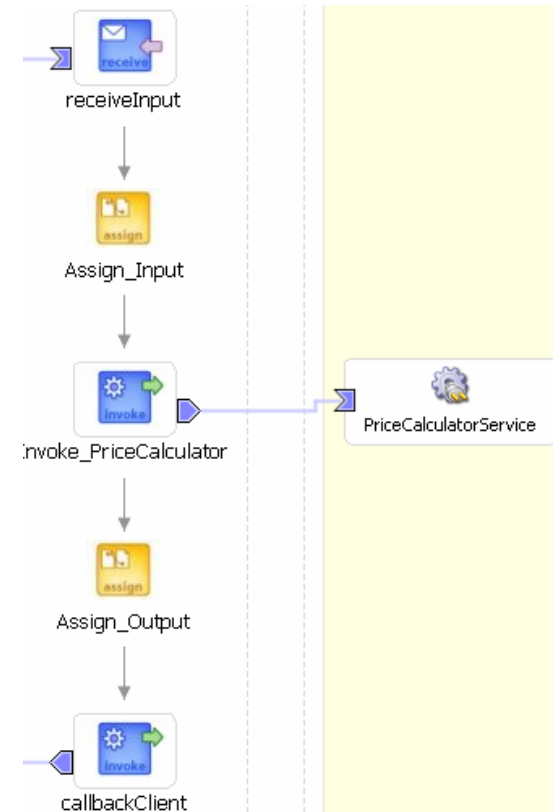
- Call a SOAP WebService
 - And wrap the Java service as a WebService
- Call a Java Service wrapped with a WSIF binding
 - For either EJB, Servlet/Http and plain (local) Java
- Use exec tag to execute (local) Java code
 - `<bpelx:exec>` to execute a snippet of Java code
 - The snippet can invoke Entity and Session EJBs
 - Not part of BPEL standard

● POJO PriceCalculator

```
public static float calculateOrderTotal
    ( int itemCount, int warrantyLevel, String product)
```

● BPEL Process GetPriceQuote

- Needs to call PriceCalculator for getting the base price
- May add some discounting logic...



- Describes the Service details for consumers

- What:

```

<service name="PriceCalculator">
  <documentation>CalculateOrderPrice</documentation>
  <plnk:partnerLinkType name="PriceCalculator">
    <plnk:role name="PriceCalculatorServiceProvider">
      <plnk:portType name="tns:PriceCalculatorPortType"/>
    </plnk:role>
  </plnk:partnerLinkType>
  <operation name="calculateOrderTotal">
    <input name="calculateOrderTotalORequest"
      message="tns:calculateOrderTotalORequest"/>
    <output name="calculateOrderTotalOResponse"
      message="tns:calculateOrderTotalOResponse"/>
  </operation>
  <message name="calculateOrderTotalORequest">
    <part name="itemCount" type="xsd:integer"/>
    <part name="warrantyLevel" type="xsd:integer"/>
    <part name="product" type="xsd:string"/>
  </message>
  <message name="calculateOrderTotalOResponse">
    <part name="return" type="xsd:float"/>
  </message>
</service>

```

- Where:

```

<port name="JavaPort" binding="tns:JavaBinding">
  <java:address className="nl.amis.sales.PriceCalculator"/>
</port>
<binding name="JavaBinding" type="tns:PriceCalculatorPortType">
  <java:binding/>

```

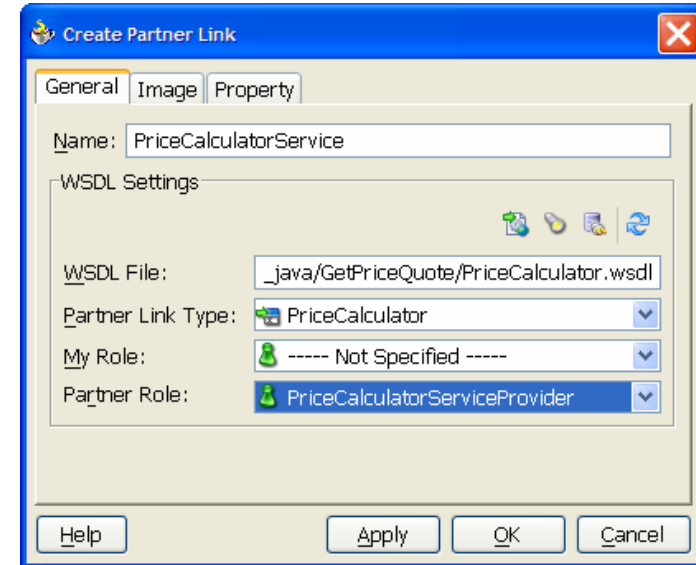
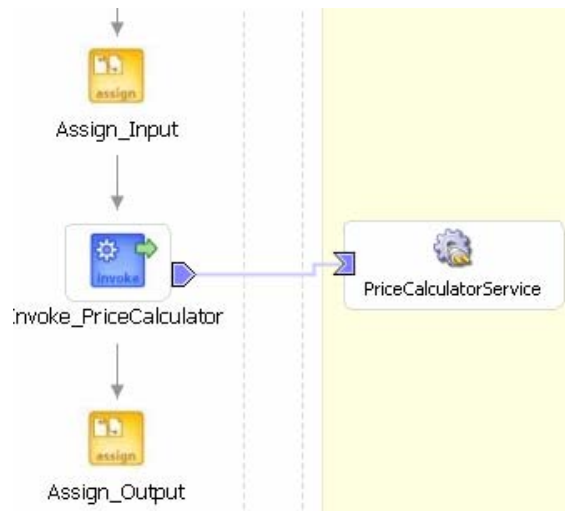
- Mapping WSDL/Java:

```

  <format:typeMapping encoding="Java" style="Java">
    <format:typeMap typeName="xsd:string" formatType="java.lang.String"/>
    <format:typeMap typeName="xsd:boolean" formatType="boolean"/>
    <format:typeMap typeName="xsd:integer" formatType="int"/>
    <format:typeMap typeName="xsd:long" formatType="long"/>
    <format:typeMap typeName="xsd:float" formatType="float"/>
  </format:typeMapping>
  <operation name="calculateOrderTotal">
    <java:operation methodName="calculateOrderTotal"/>
  </operation>
</binding>

```

- Create PartnerLink
 - Browse for WSIF WSDL
- From here on there is no difference between Java & WSIF based services and other services



- Note: the Java Classes need to be available on the BPEL PM Server

BPEL Process: **GetPriceQuote** Version: 1.0 Lifecycle: **ORACLE** BPEL Console Manage BPEL Domain | Log

Statistics: [0 Open Instances](#) | [9 Closed Instances](#)

[Manage](#) **Initiate** [Descriptor](#) [WSDL](#)

Testing this BPEL Process

Initiating a test instance HTML Form

To create a new 'test' instance of this BPEL Process, fill this form

GetPriceQuoteProcessRequest

product	TV
numberOfItems	5
warrantyLevel	4

Dashboard **BPEL Processes** **Instances**

Title: Instance #713 of GetPriceQuote Last Modified: 6/9/06 9:
 Reference Id: 713 Tree Finder State: closed.co
 BPEL Process: [GetPriceQuote \(v. 1.0\)](#) Priority: 3

[Manage](#) **Flow** [Audit](#) [Debug](#) [Interactions](#)

Visual representation of the history of this BPEL business flow [As of 6/9/06 9:04:16 AM]

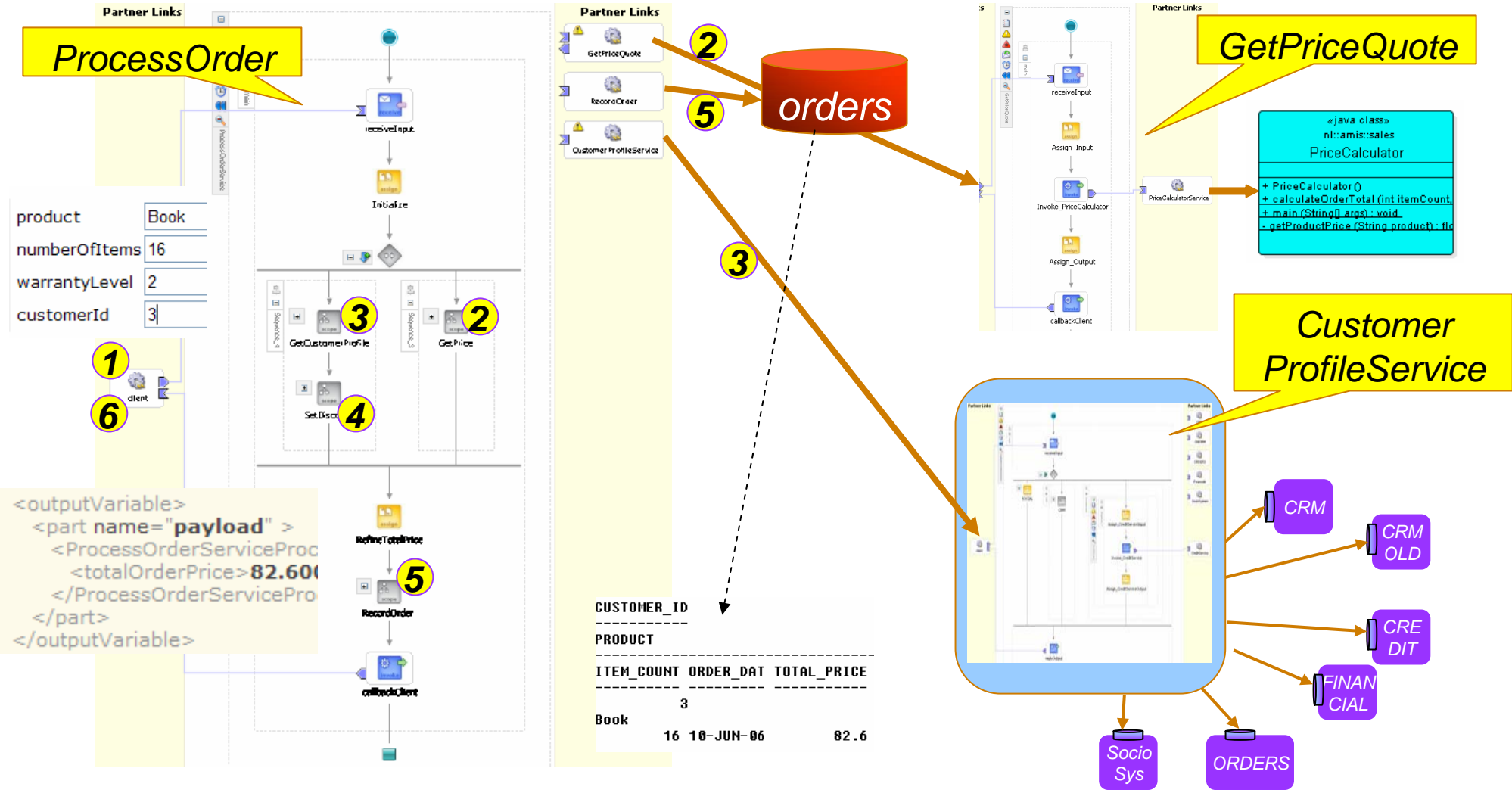
Invoke_PriceCalculator

[2006/06/09 09:04:14]
 Invoked 2-way operation "calculateOrderTotal" on partner "PriceCalculatorService".

```
<messages>
  <Invoke_1_calculateOrderTotal_InputVariable>
    <part name="product" >
      <product>TV</product>
    </part>
    <part name="itemCount" >
      <itemCount>5</itemCount>
    </part>
    <part name="warrantyLevel" >
      <warrantyLevel>4</warrantyLevel>
    </part>
  </Invoke_1_calculateOrderTotal_InputVariable>
  <Invoke_1_calculateOrderTotal_OutputVariable>
    <part name="return" >
      <return>9895.1</return>
    </part>
  </Invoke_1_calculateOrderTotal_OutputVariable>
</messages>
```



- The ProcessOrder service
 - ① ■ takes an order for a specific customer
 - ② ■ gets a price quote from the GetPriceQuote service
 - ③ ■ gets the customer profile from the CustomerProfileService
 - ④ ■ awards a discount depending on the CustomerStatus
 - ⑤ ■ processes the order straight into the Orders Database
 - ⑥ ■ returns the total price for the order



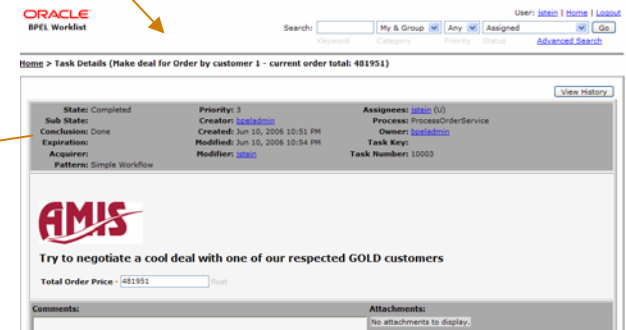
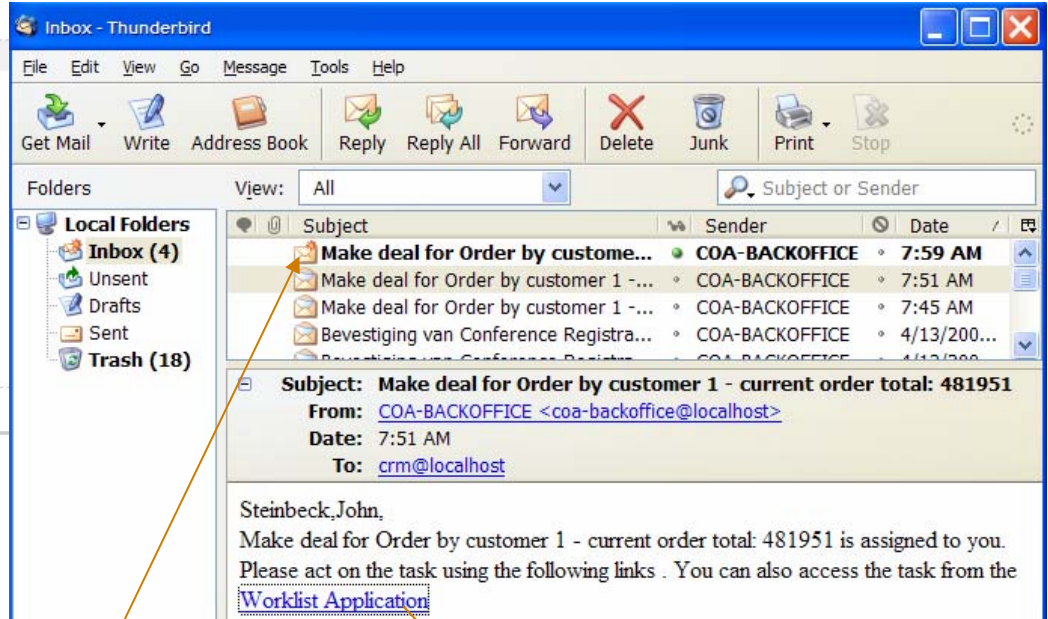
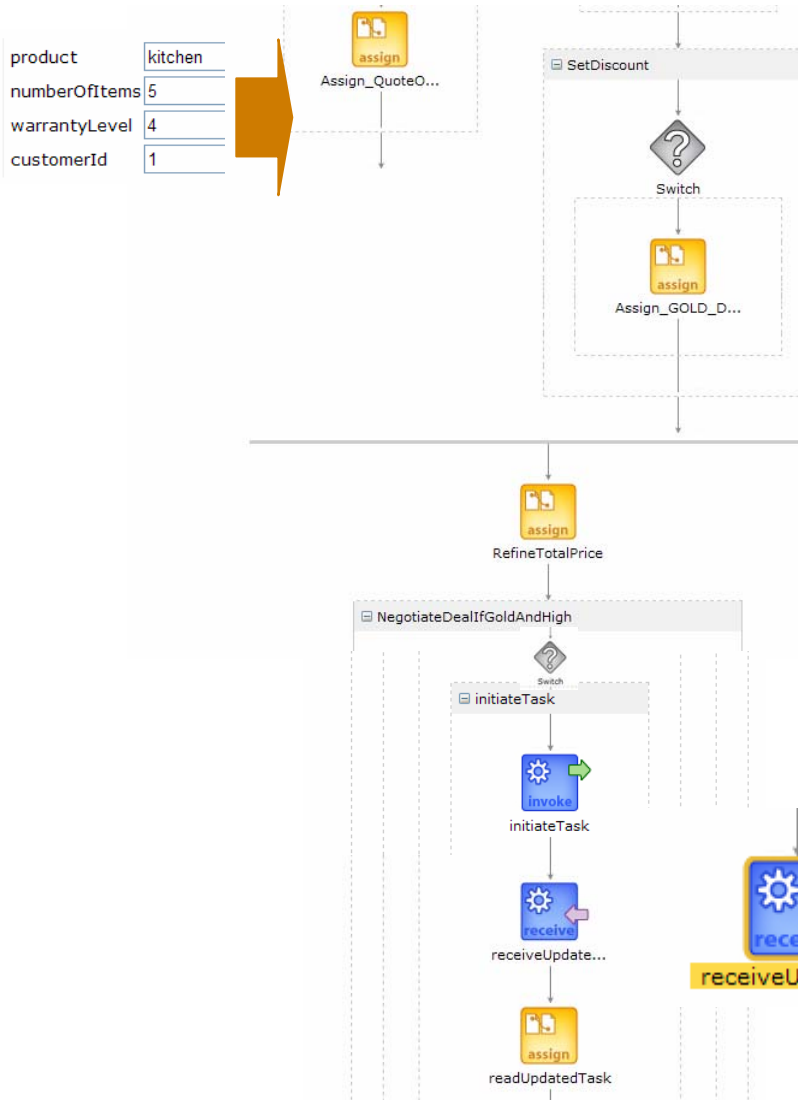
- Introduction of Service Oriented Architecture and BPEL (Business Process Execution Language)
- *Calling BPEL Services from Java Applications*
 - When, Why and How?
- *Invoking Java based Services from a BPEL Process*
 - When, Why and How?
- **Workflow Tasks picked up and performed by Java applications**
 - Human Workflow and Complex Automated Workflow
- Conclusions

- Many Business Processes require a form of *humanual* activities
 - choose, approve, confer, communicate
- BPEL Processes often contain such humanual steps
 - in addition to automated service calls
- A User Interface is typically required for the human workflow steps
 - Email to notify
 - Worklist to presents the outstanding tasks
 - (Web) User Interface to inspect the task details, perform the required operations and complete the task
- Java Web Applications are very well suited for providing this Humanual Task User Interface

- If order total is over 10k and the customer status is GOLD, let's try to negotiate a deal
 - Start workflow in which a human needs to negotiate a deal, update the total price and complete the workflow
 - An email is sent for notification of the employee to start working on the deal-task
 - A Java Web UI is used to present the task details and allow update of Total Price and Task Status
 - When the Workflow completes, the ProcessOrder process resumes, possibly with an updated TotalPrice



ProcessOrder BPEL Process with Workflow





Very slightly modified Generic Oracle BPEL PM Worklist Application

ORACLE
BPEL Worklist

User: [jstein](#) | [Home](#) | [Logout](#)

Search:

Keyword Category: Priority: Status:

[Home](#) > Task Details (Make deal for Order by customer 1 - current order total: 481951)

[View History](#)

State: Completed	Priority: 3	Assignees: jstein (U)
Sub State:	Creator: bpeladmin	Process: ProcessOrderService
Conclusion: Done	Created: Jun 10, 2006 10:51 PM	Owner: bpeladmin
Expiration:	Modified: Jun 10, 2006 10:54 PM	Task Key:
Acquirer:	Modifier: jstein	Task Number: 10003
Pattern: Simple Workflow		



Try to negotiate a cool deal with one of our respected GOLD customers

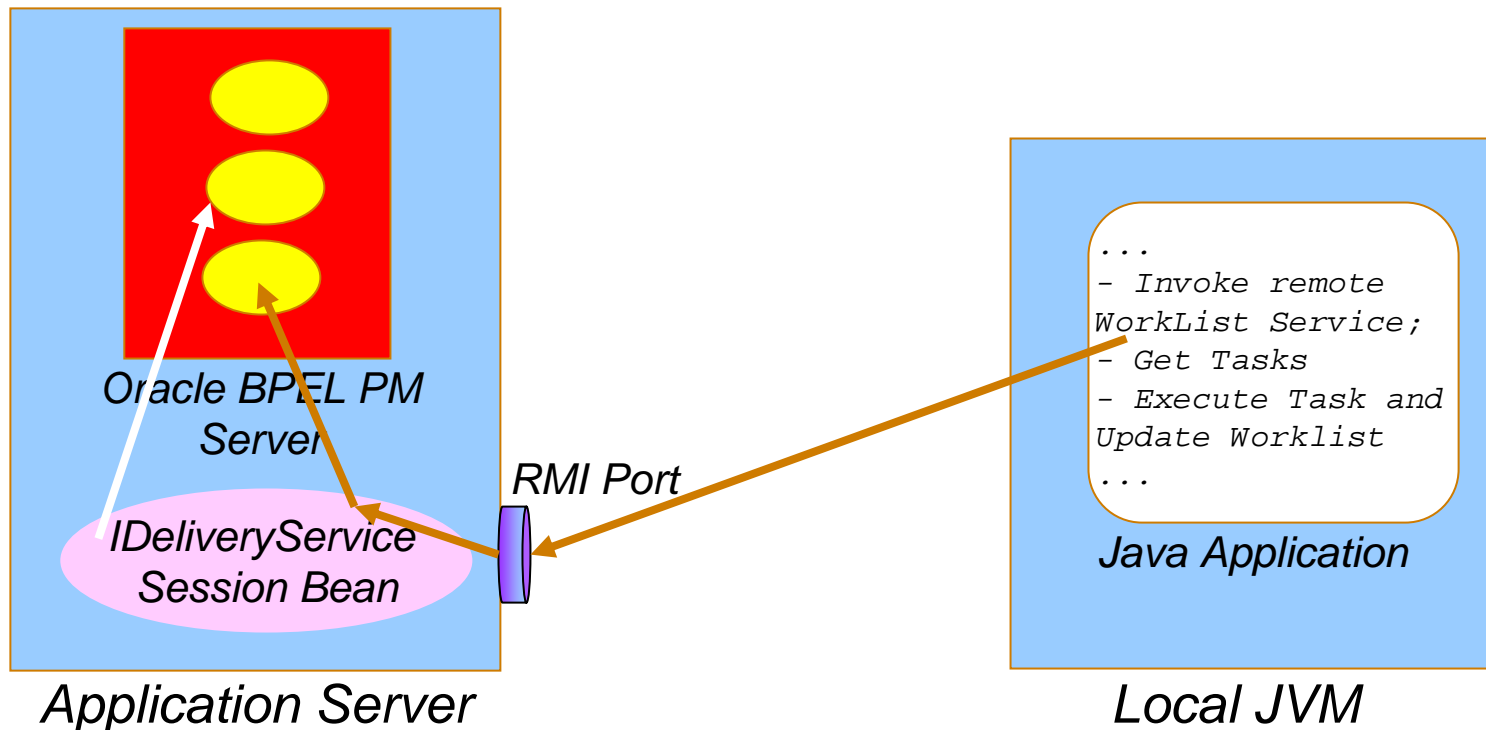
Total Order Price • float

Comments:

Attachments:

No attachments to display.

- Our Java Application can implement the interface for (human) workflow activities in BPEL processes
 - Java GUI (Swing), Java Web App, Java WebService can invoke the WorkList Service



```
// 1. get a handle to the remove worklist service client
RemoteWorklistServiceClient client = new RemoteWorklistServiceClient();
client.init();
// 2. get worklist context for user
IWorklistContext ctx = client.authenticateUser("jstein", "welcome");
// 3. set filters for retrieving My tasks with Assigned status
Map filterMap = new HashMap();
filterMap.put(IWorklistService.FILTER_TYPE_TASK_FILTER,
             IWorklistService.TASK_FILTER_MY);
filterMap.put(IWorklistService.FILTER_TYPE_STATUS_FILTER,
             IWorklistService.STATUS_FILTER_ASSIGNED);
List tasks =
    client.getWorklistTasks(ctx, filterMap, IWorklistService.SORT_FIELD_TASK_TITLE,
                          IWorklistService.SORT_ORDER_ASCENDING);
if (tasks != null) {
    for (int i = 0; i < tasks.size(); i++) {
        Task task = (Task)tasks.get(i);
        String taskId = task.getTaskId();
        System.out.println(task.getTitle());
    }
}
```

```
Connecting to WorklistService as jstein
```

```
Got Worklist Context
```

```
Make deal for Order by customer 1 - current order total: 27301
```

```

Element payload = (Element)task.getPayload();
float totalPrice = Float.parseFloat(payload.getNodeValue());
// automatically award another 13% discount
totalPrice = (float)(totalPrice * 0.87);
payload.setNodeValue(Float.toString(totalPrice));
client.updateTask(ctx, task);
// add some comments to the task
client.appendTaskComments(ctx, taskId,
    "Programmatically manipulated by nl.amis.sales.crm.DealNegotiatorClient on " +
    new java.util.Date()+ ". New total price set at " +Float.toString(totalPrice));
System.out.println(
    "Programmatically manipulated by nl.amis.sales.crm.DealNegotiatorClient on " +
    new java.util.Date()+ ". New total price set at " +Float.toString(totalPrice));
// close the task by setting its status to DONE
client.customTaskOperation(ctx, taskId, "DONE");

```

```

Programmatically manipulated by nl.amis.sales.crm.DealNegotiatorClient
on Sun Jun 11 12:55:17 CEST 2006. New total price set at 23751.87

```

- Introduction of Service Oriented Architecture and BPEL (Business Process Execution Language)
- *Calling BPEL Services from Java Applications*
 - When, Why and How?
- *Invoking Java based Services from a BPEL Process*
 - When, Why and How?
- Workflow Tasks picked up and performed by Java applications
 - Human Workflow and Complex Automated Workflow
- **Conclusions**

- SOA in general and BPEL in particular promote reuse, loosely coupling, business agility
- Java Applications can easily invoke BPEL Services
 - Through RMI or SOAP WS
- BPEL Process and call Java based services
 - Through WSIF (POJO, EJB, Http/Servlet) or SOAP WS
- Java Applications can provide the User Interface for Humanual steps in BPEL Processes
- Java and BPEL – a good team!