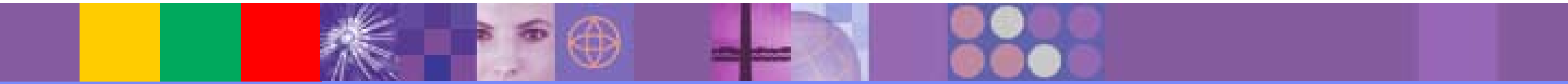
IBM Software Group

2006 B2B Customer Conference
B2B – Catch the Next Wave

B5: Service Oriented Architecture (SOA)

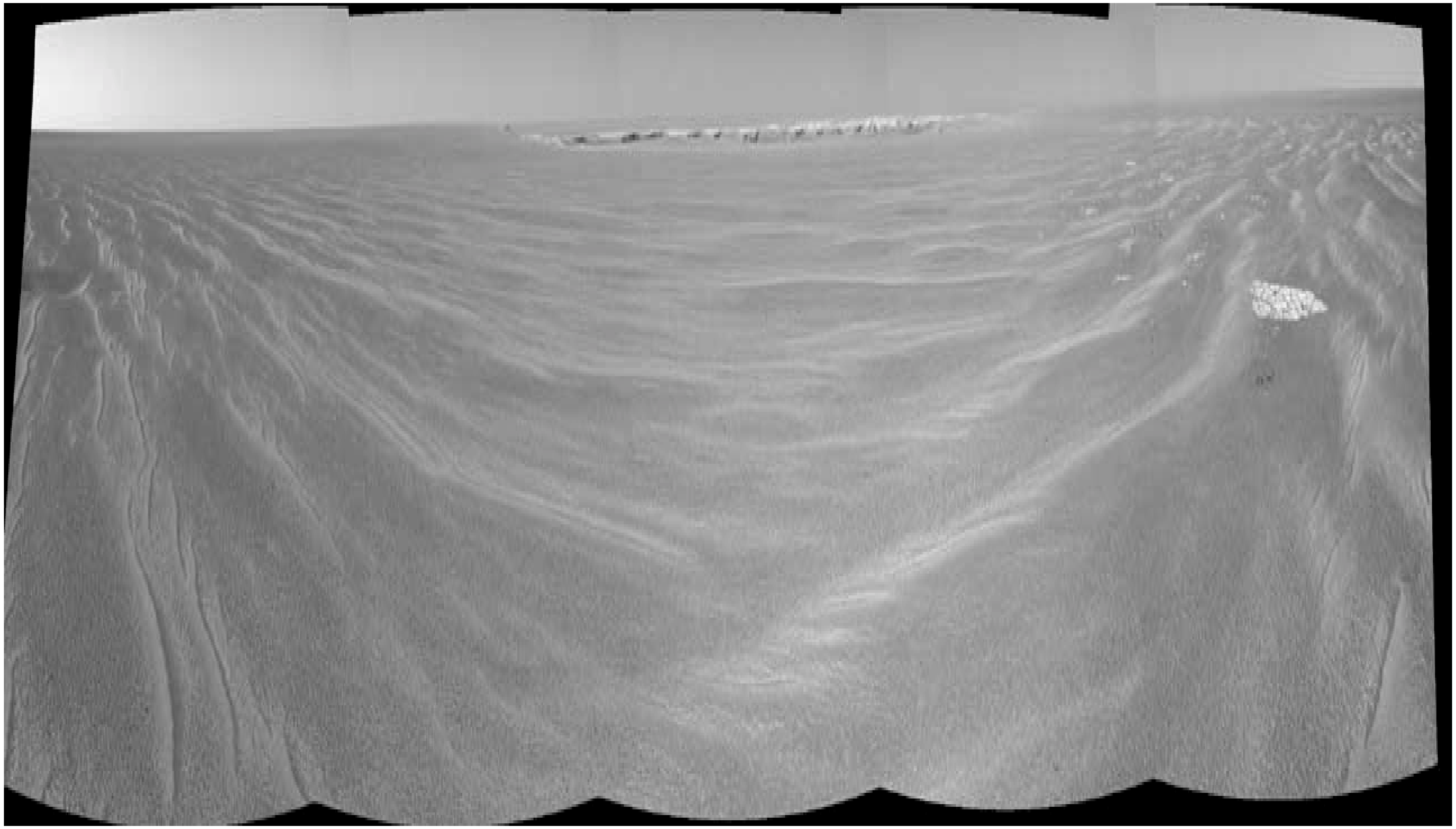
David Hixon, IBM B2B Architect

WebSphere. software



ON DEMAND BUSINESS

Introduction and Opening



Objectives

- Define Service Oriented Architecture (SOA)
- Explain the core components of SOA
  - Service Component Architecture (SCA) Programming Model
  - Service Data Objects (SDO)
  - Common Event Infrastructure (CEI)
- Show how to use WDI in a SOA environment
- Demonstrate WDI being used in a SOA environment

Introduction to SOA

- SOA is a **framework** that combines individual business functions and processes, called services, to implement sophisticated business applications and processes.
- SOA is an approach to IT that considers business processes as reusable components or services which are **loosely-coupled** and that are platform and implementation **neutral**.
- The solution can then be viewed as a **composite application** consisting of a choreographed set of service interactions defined by graphically wiring together the svcs
- The approach allows you to design solutions as assemblies of services in which the assembly description is a managed, **well-defined first-class aspect** of the solution, and hence, amenable to analysis, change, and evolution.

Core Ideas of SOA

- **Service** – A service is a logical unit of functionality that can be used across applications
- **Access** (SCA) – the heart of SOA is a common way to access and describe services
  - Directory
  - Transport
  - Interface
- **Information** (SDO) – a common way to access data
  - Parsing and serialization
  - Meta data
  - Navigation

Core Ideas of SOA (*cont.*)

- **Events** – a common way to monitor applications and handle alerts
  - *Common Business Event* (CBE) A consistent specification for the definition of normalized event and log information for various domains (business, security, network, system, etc.)
    - Value: Richer and normalized data enables cross-product analysis & correlation; is a prerequisite to effective root cause analysis and automation
  - *Common Event Infrastructure* (CEI) - A readily available, reliable, scalable and embeddable event infrastructure that supports submission, persistence and distribution of event data based on CBE/WEF through standard APIs so that events can be shared for management purposes
    - Value: Robust event infrastructure facilitates exchange of information among cross domains event producers and consumers for real time management purposes

A Simple Example

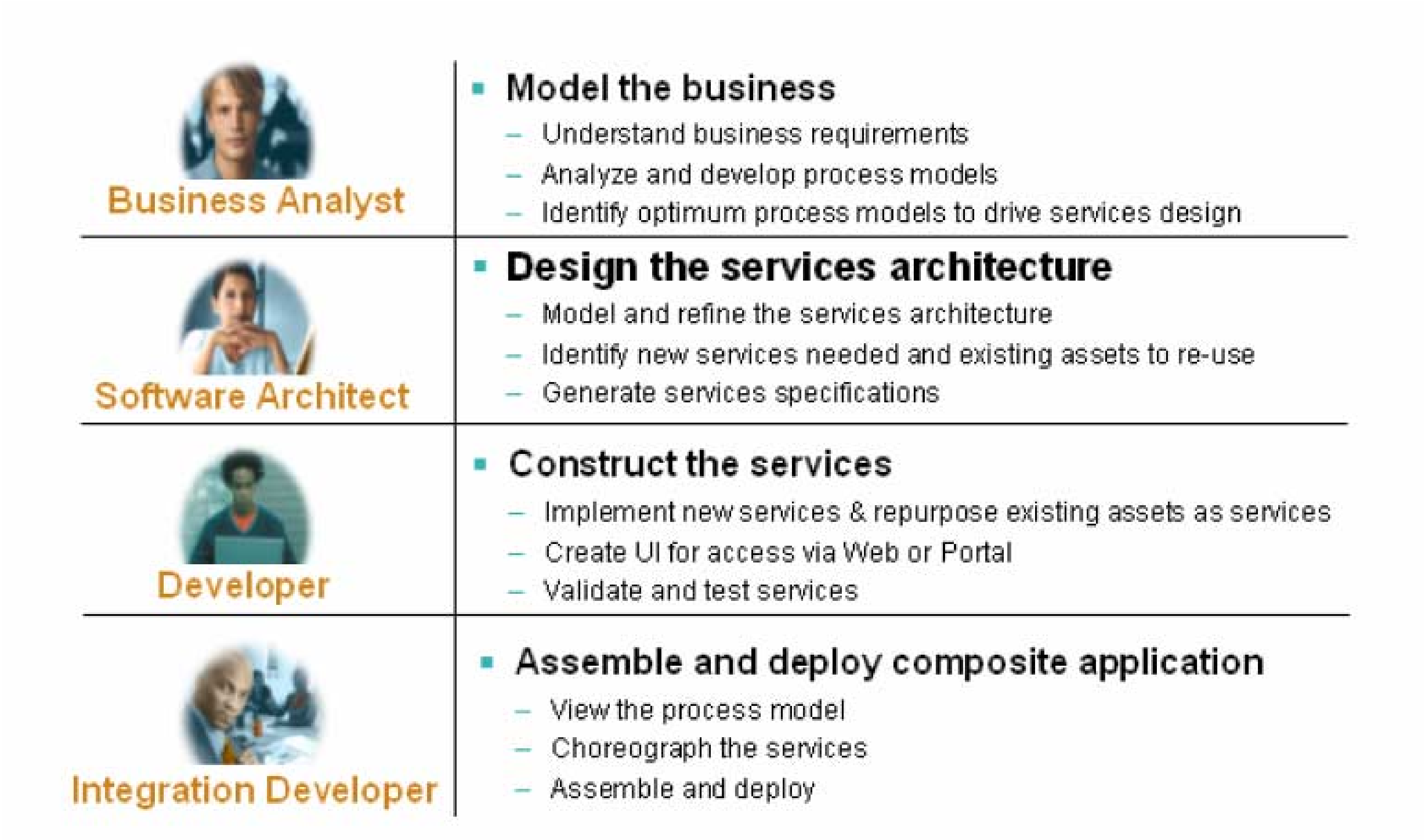
- Create a web application that draws information and services from the following sources
  - CICS

    LU6.2? COMM areas?
  - SAP/R3

    Sockets? ABAPI?
  - C++

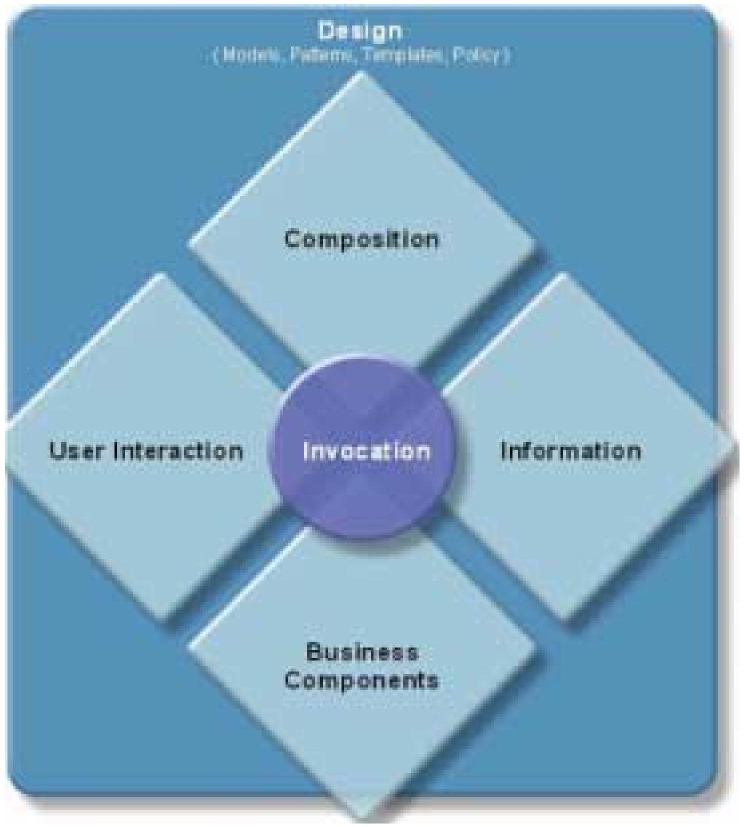
    MQ? Native C++ app?
  - EJB
- If the business functionality of each system is exposed as a service, then creating a composite application is simple.

Key Roles in Service Oriented Design/Dev

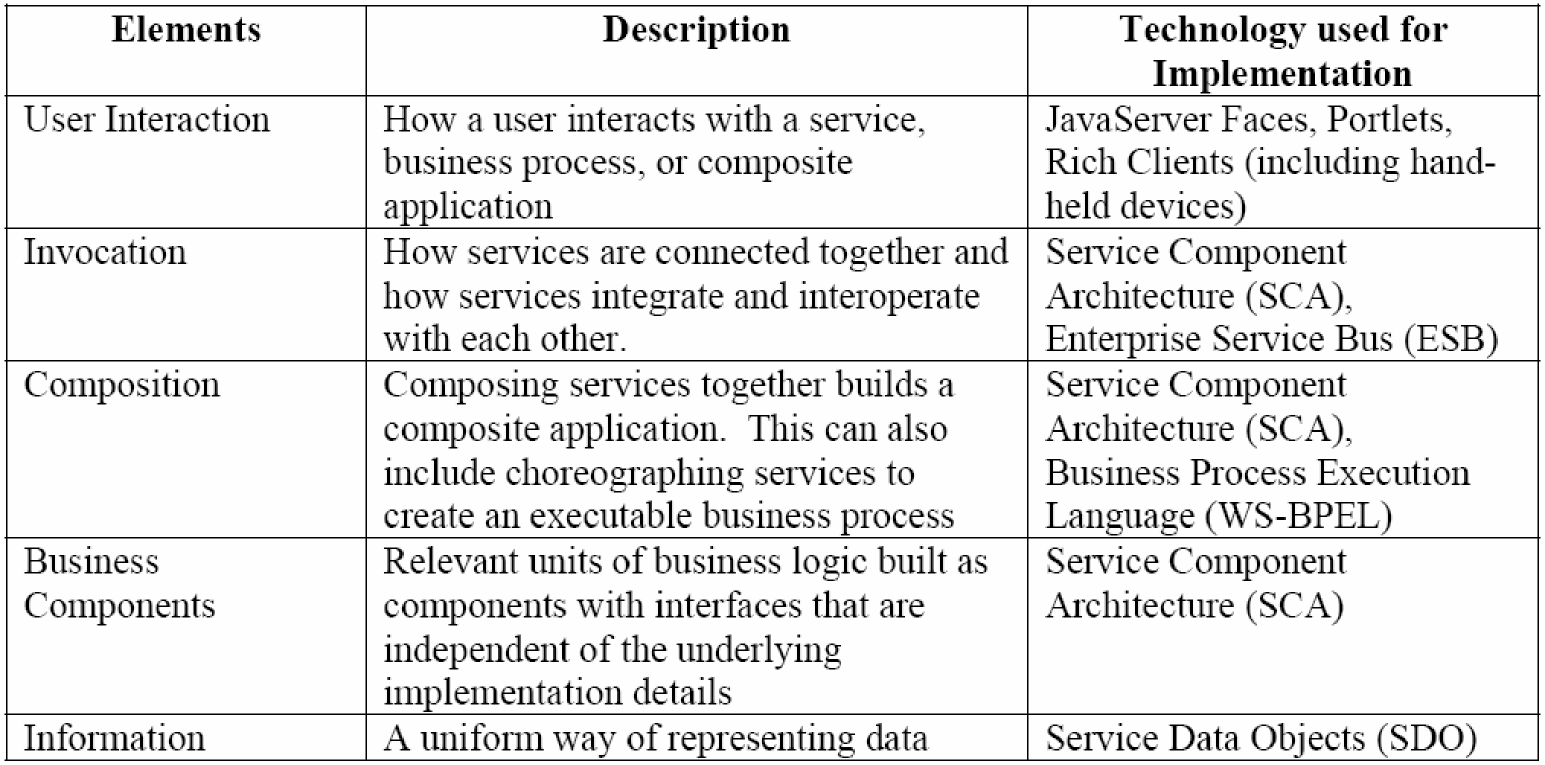


Business Analyst	▪ **Model the business** – Understand business requirements – Analyze and develop process models – Identify optimum process models to drive services design
Software Architect	▪ **Design the services architecture** – Model and refine the services architecture – Identify new services needed and existing assets to re-use – Generate services specifications
Developer	▪ **Construct the services** – Implement new services & repurpose existing assets as services – Create UI for access via Web or Portal – Validate and test services
Integration Developer	▪ **Assemble and deploy composite application** – View the process model – Choreograph the services – Assemble and deploy

SOA Programming Model Elements

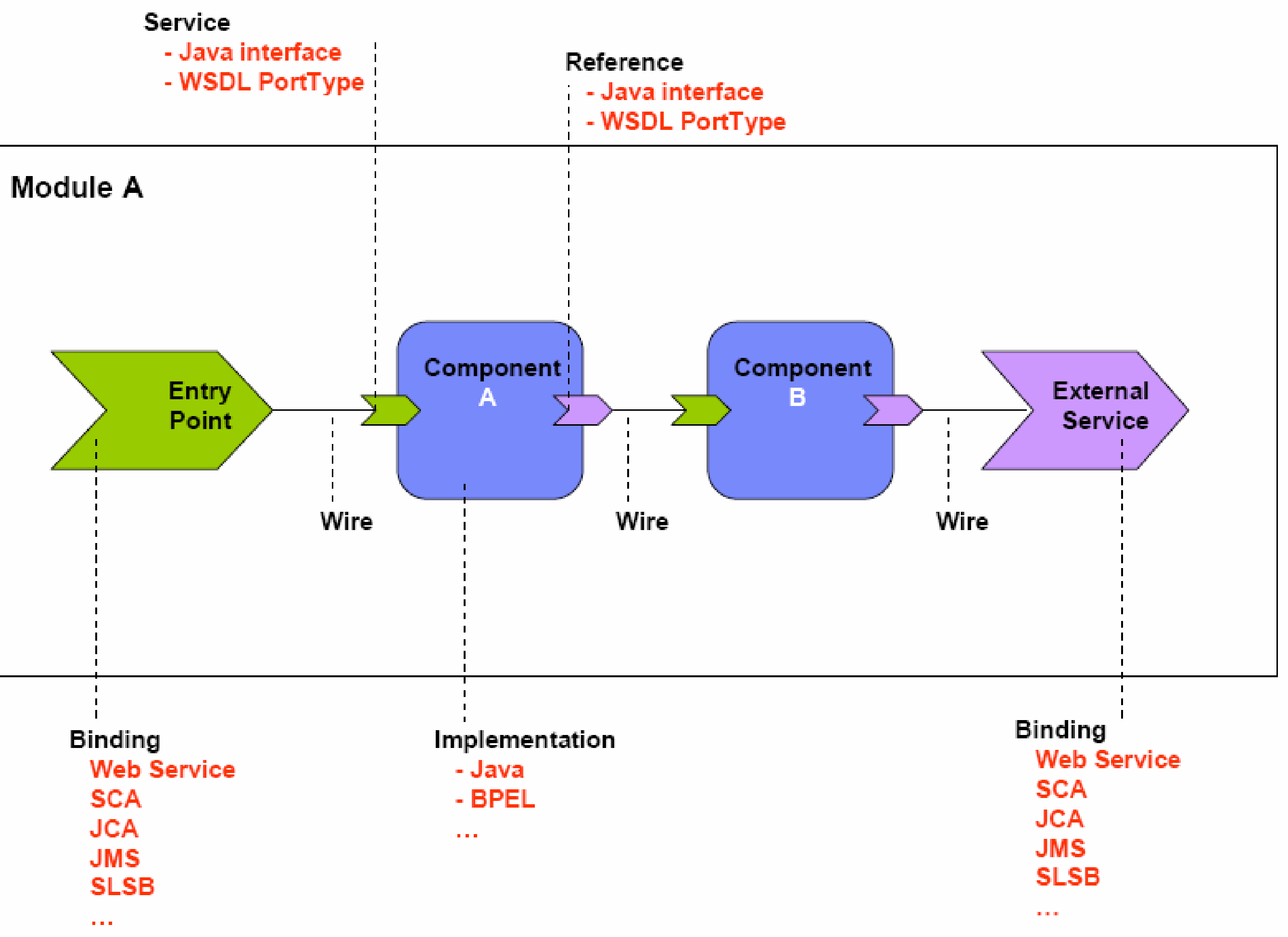


SOA Programming Model Elements

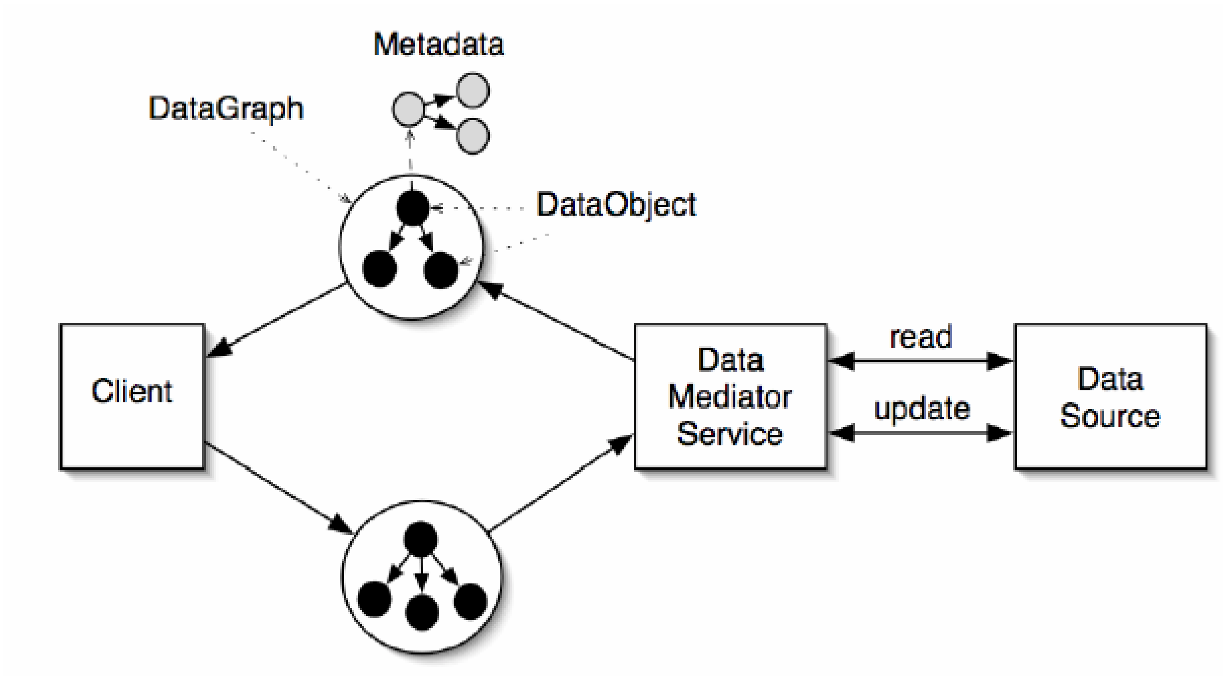


Elements	Description	Technology used for Implementation
User Interaction	How a user interacts with a service, business process, or composite application	JavaServer Faces, Portlets, Rich Clients (including hand-held devices)
Invocation	How services are connected together and how services integrate and interoperate with each other.	Service Component Architecture (SCA), Enterprise Service Bus (ESB)
Composition	Composing services together builds a composite application. This can also include choreographing services to create an executable business process	Service Component Architecture (SCA), Business Process Execution Language (WS-BPEL)
Business Components	Relevant units of business logic built as components with interfaces that are independent of the underlying implementation details	Service Component Architecture (SCA)
Information	A uniform way of representing data	Service Data Objects (SDO)

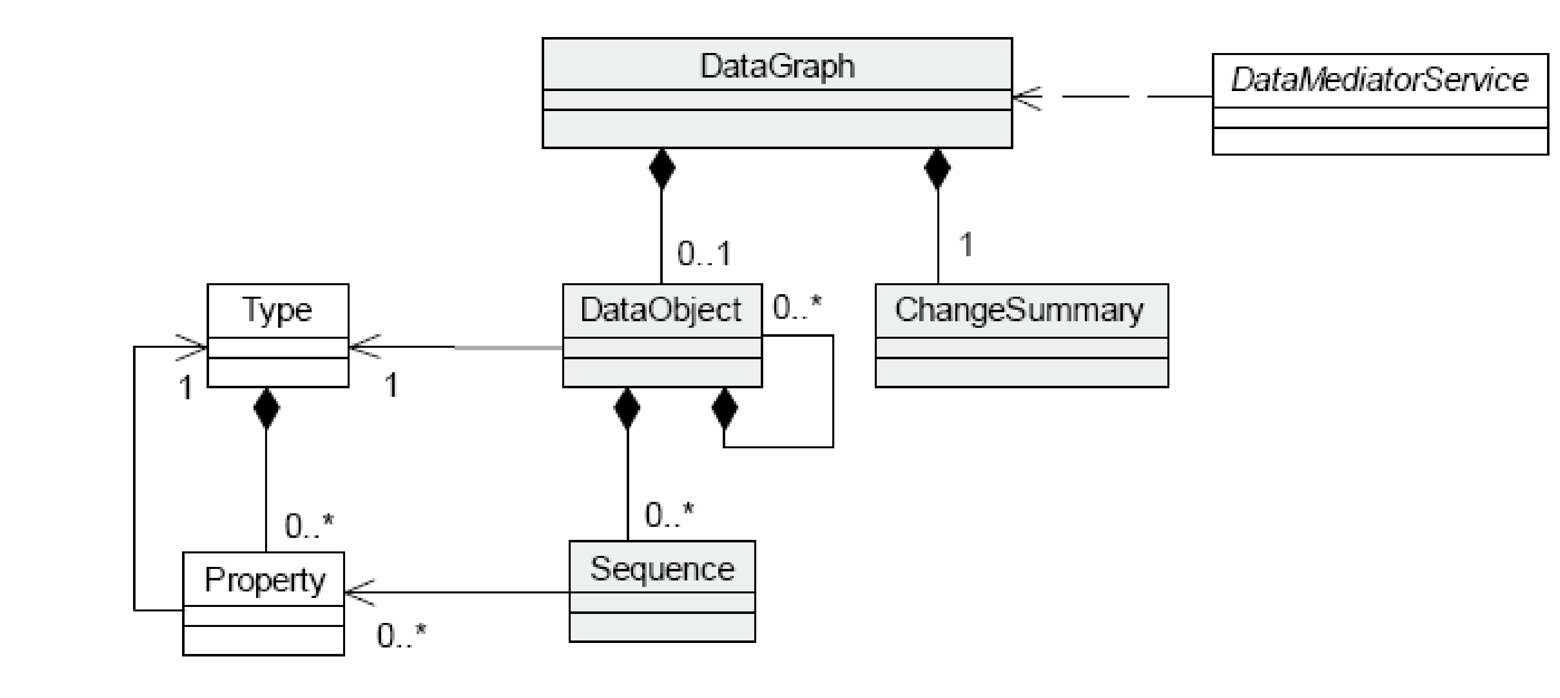
SCA - A Simple Module



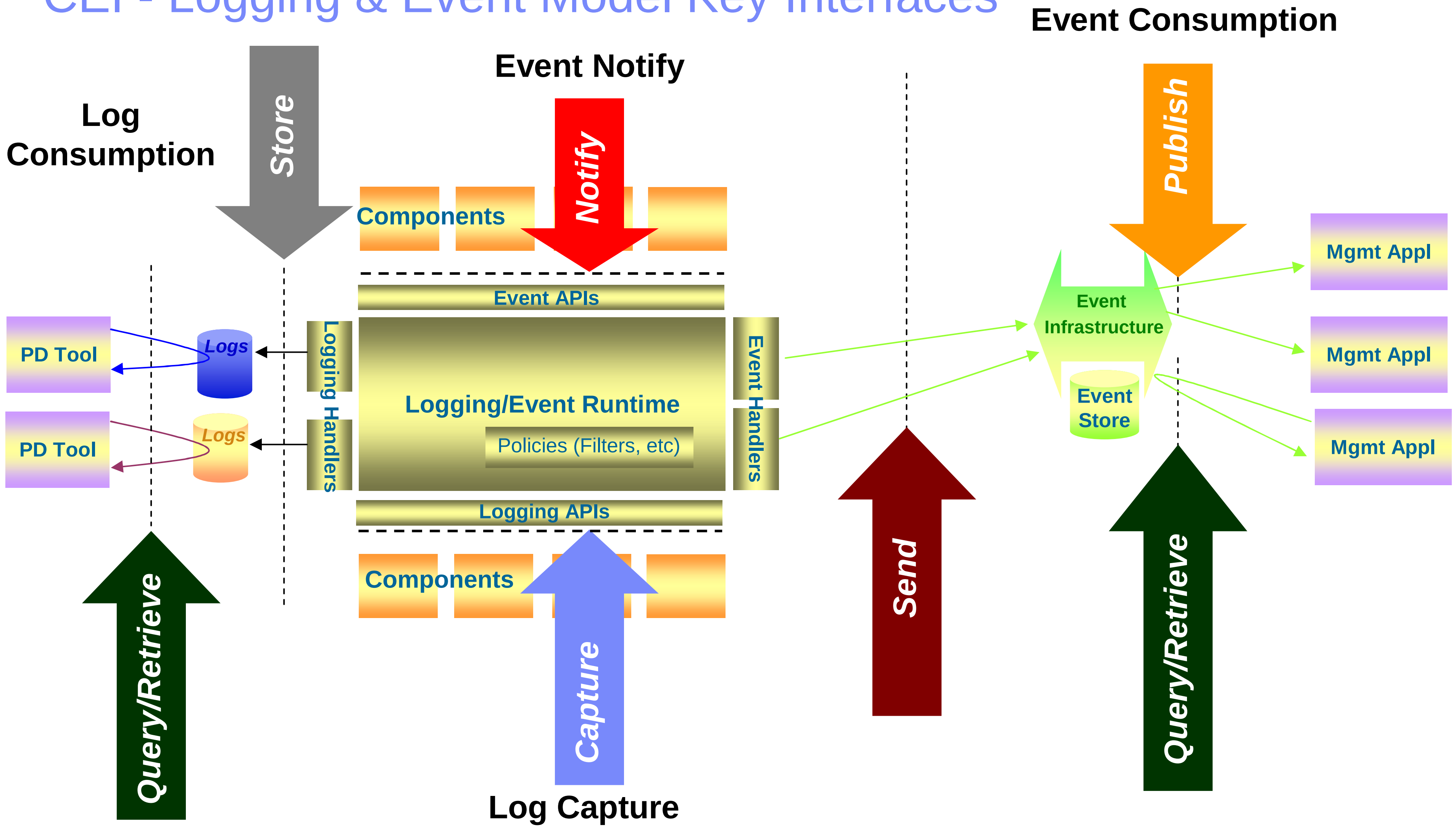
SDO - Components of an SDO Solution



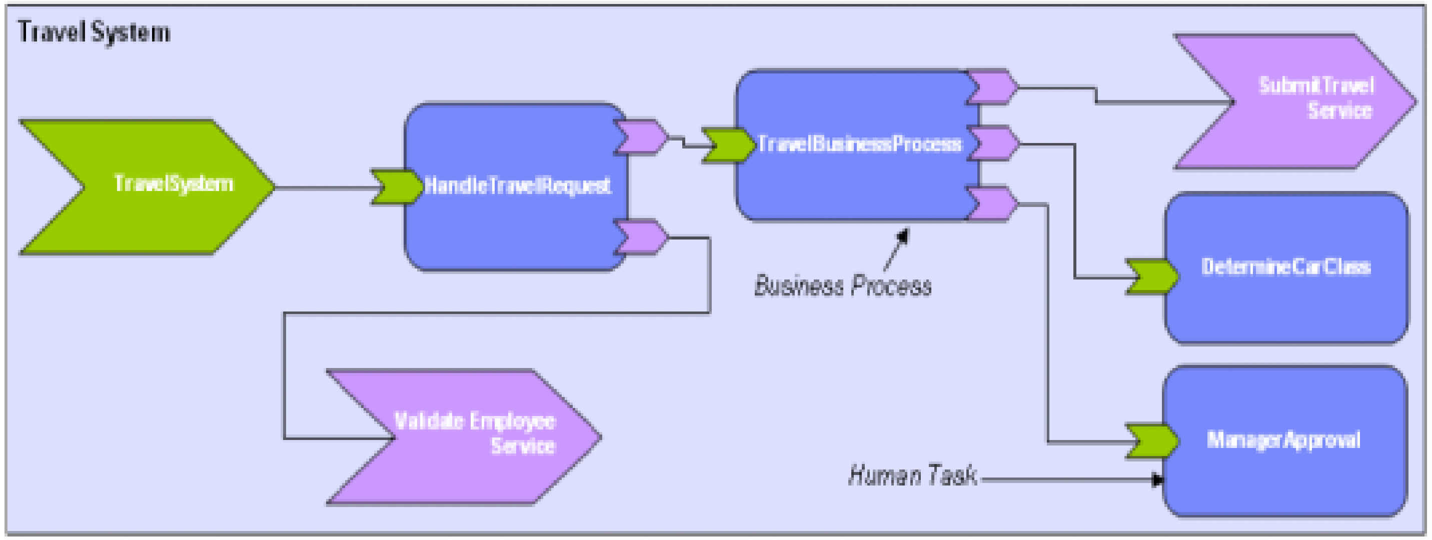
UML Model of Core SDO Components

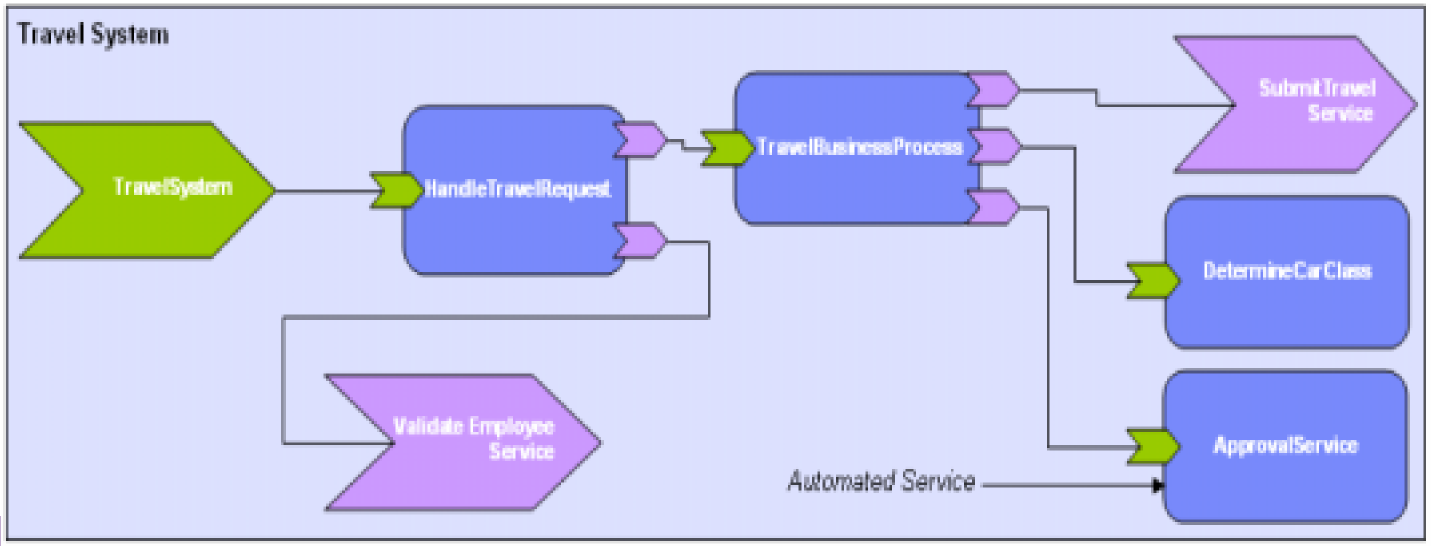


CEI - Logging & Event Model Key Interfaces

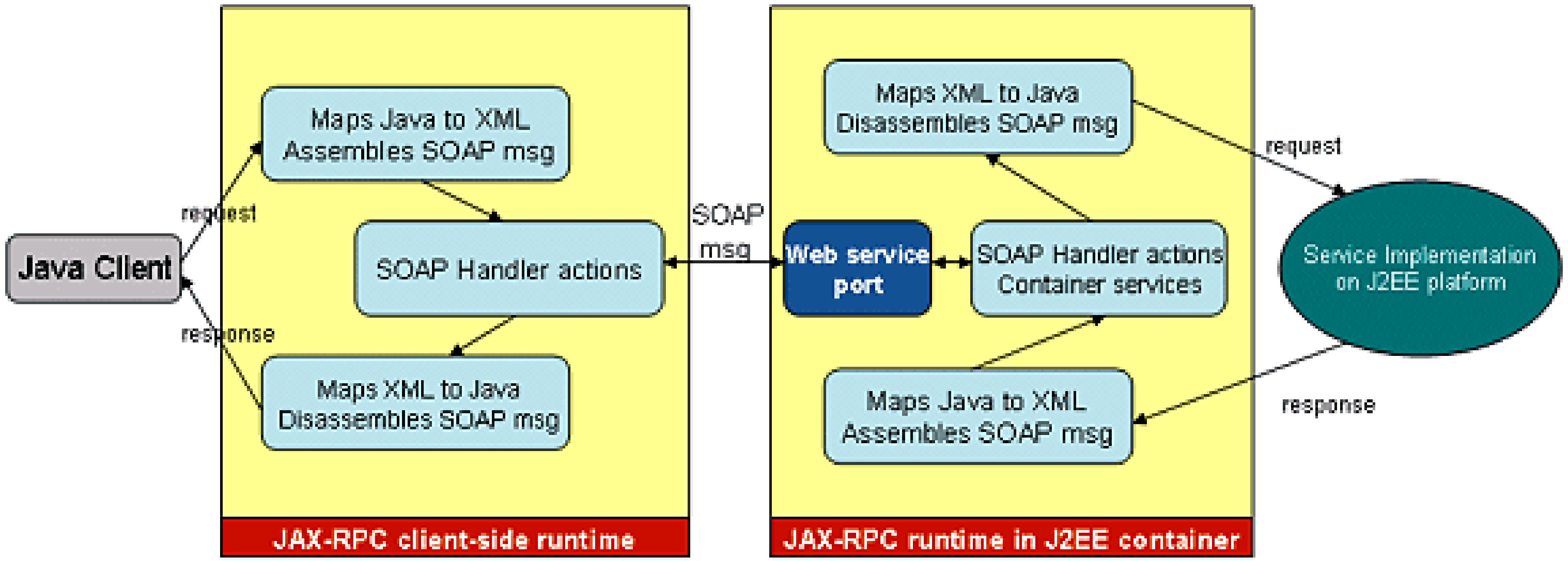


Example of rewiring an assembly

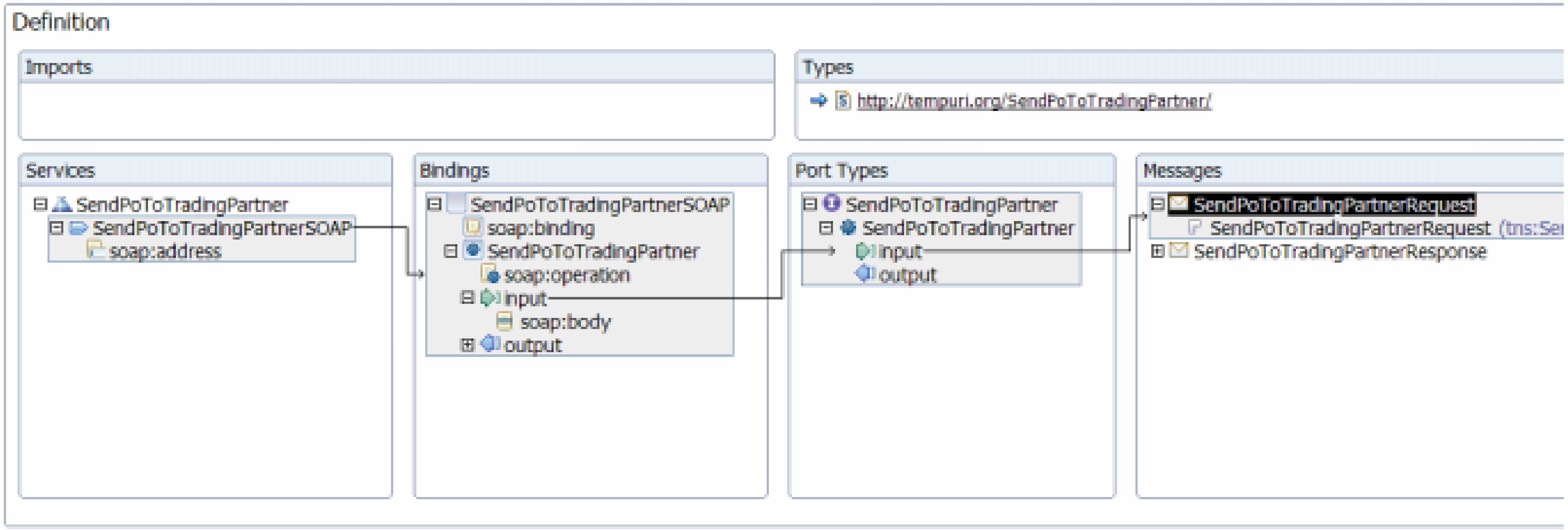




Web Services Based SOA



Generate a WSDL Using the WSDL Wizard


Definition

Imports

Types
http://tempuri.org/SendPoToTradingPartner/

Services
SendPoToTradingPartner
SendPoToTradingPartnerSOAP
soap:address

Bindings
SendPoToTradingPartnerSOAP
soap:binding
SendPoToTradingPartner
soap:operation
input
soap:body
output

Port Types
SendPoToTradingPartner
SendPoToTradingPartner
input
output

Messages
SendPoToTradingPartnerRequest
SendPoToTradingPartnerRequest (tns:Se
SendPoToTradingPartnerResponse

Send PO to Trading Partner Generated WSDL Source

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://tempuri.org/SendPoToTradingPartner/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="SendPoToTradingPartner"
    targetNamespace="http://tempuri.org/SendPoToTradingPartner/">
  <wsdl:types>
    <xsd:schema targetNamespace=http://tempuri.org/SendPoToTradingPartner/
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="SendPoToTradingPartnerResponse" type="xsd:string"/>
      <xsd:element name="SendPoToTradingPartnerRequest" type="xsd:string"/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="SendPoToTradingPartnerResponse">
    <wsdl:part element="tns:SendPoToTradingPartnerResponse"
    name="SendPoToTradingPartnerResponse"/>
  </wsdl:message>
  <wsdl:message name="SendPoToTradingPartnerRequest">
    <wsdl:part element="tns:SendPoToTradingPartnerRequest"
    name="SendPoToTradingPartnerRequest"/>
  </wsdl:message>
```

Generated WSDL Source (*cont.*)

```
 <wsdl:portType name="SendPoToTradingPartner">
   <wsdl:operation name="SendPoToTradingPartner">
     <wsdl:input message="tns:SendPoToTradingPartnerRequest"/>
     <wsdl:output message="tns:SendPoToTradingPartnerResponse"/>
   </wsdl:operation>
  </wsdl:portType>
 <wsdl:binding name="SendPoToTradingPartnerSOAP"
   type="tns:SendPoToTradingPartner">
   <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="SendPoToTradingPartner">
     <soap:operation
   soapAction="http://tempuri.org/SendPoToTradingPartner/NewOperation"/>
     <wsdl:input> <soap:body use="literal"/> </wsdl:input>
     <wsdl:output> <soap:body use="literal"/> </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SendPoToTradingPartner">
   <wsdl:port binding="tns:SendPoToTradingPartnerSOAP"
   name="SendPoToTradingPartnerSOAP">
     <soap:address location="http://tempuri.org"/>
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Generate a Java Bean from the WSDL

```
/ * This file was auto-generated from WSDL
 *   by the IBM Web services WSDL2Java emitter.
 *   o0526.04 v62905175048
 */
package org.tempuri;

public class SendPoToTradingPartnerSOAPImpl implements
      org.tempuri.SendPoToTradingPartner_PortType
{
   public java.lang.String sendPoToTradingPartner(java.lang.String sendPoToTradingPartnerRequest)
     throws java.rmi.RemoteException
   {
      return null;
   }
}
```

Fill out the Skeleton Java Bean

```
 / * This file was auto-generated from WSDL
 *   by the IBM Web services WSDL2Java emitter.
 *   o0526.04 v62905175048
 */
package org.tempuri;
import java.util.Properties;
import java.nio.ByteBuffer;
import com.ibm.wdi.Translator;
public class SendPoToTradingPartnerSOAPImpl implements
   org.tempuri.SendPoToTradingPartner_PortType{
   public java.lang.String sendPoToTradingPartner(java.lang.String sendPoToTradingPartnerRequest)
   throws java.rmi.RemoteException {
   try {
      // Put the name of the service profile to use in the properties object:
      Properties msgProperties = new Properties();
      msgProperties.put("SVCPROF", "SENDPO");

      // Send to the transform service:
      Translator myTransformService = new Translator();
      myTransformService.invokeServiceProfile(msgProperties, sendPoToTradingPartnerRequest);

      // Nothing to return in this scenario:
      return "SUCCESS";
   }
   catch (Exception e) {
      throw new java.rmi.RemoteException("Transformation exception.  See WDI print file.", e);
   }
}
```

The WDI POJO invokeServiceProfile Service

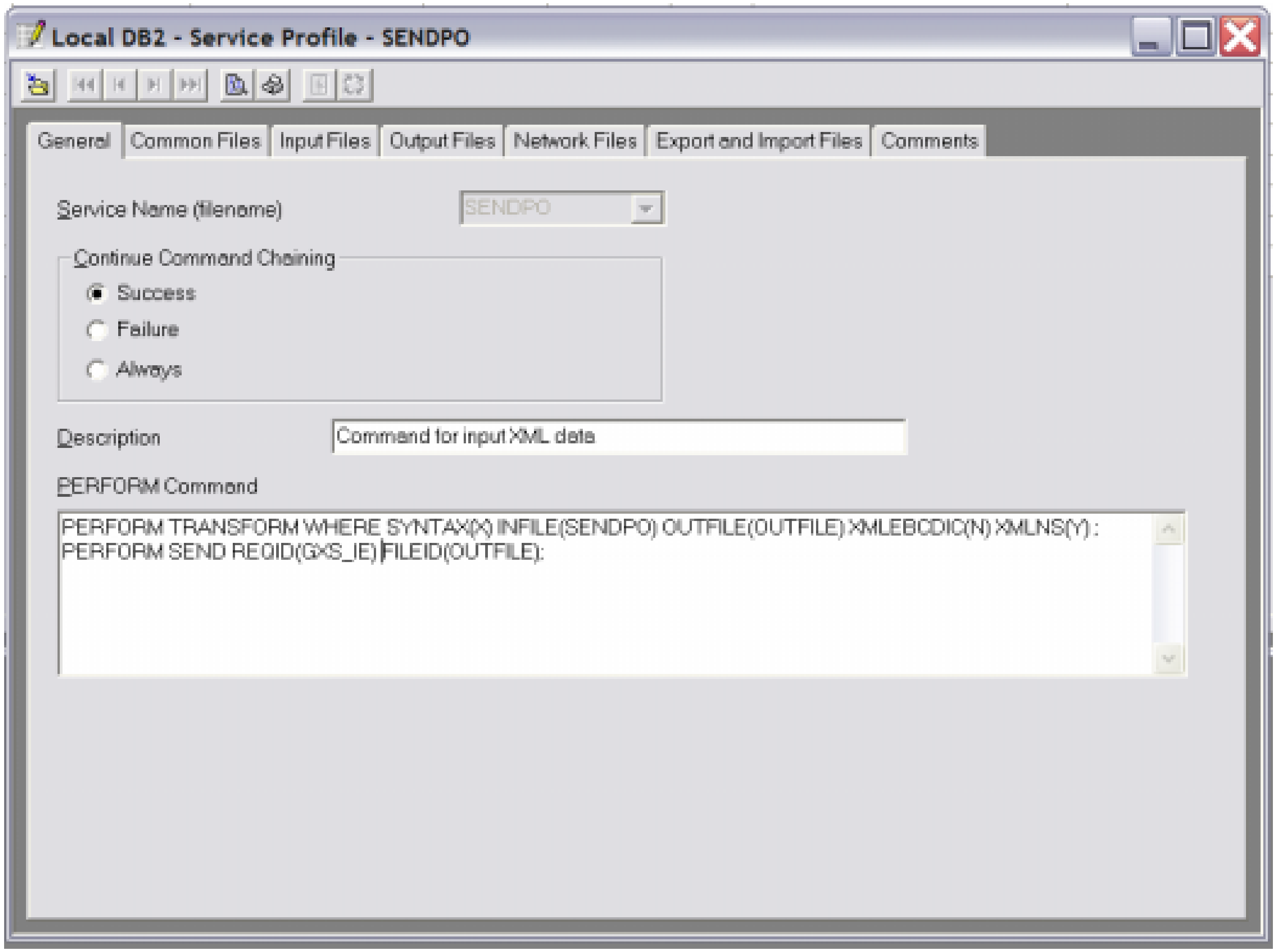
```
public void invokeServiceProfile(Properties MsgProperties, java.lang.String srcMsg)
  throws WdiTransformException {
  try {
    // Create a temporary input file and copy in source message:
    String serviceProfile = MsgProperties.getProperty(“SVCPROF”);
    inFile = new File(serviceProfile + “.in");
    FileWriter out = new FileWriter(inFile );
    out.write(srcMsg );
    out.close();

   // Name the input and print files for WDI:
   wdiTranslator.setFileName(serviceProfile , serviceProfile + “.in");
   wdiTranslator.setFileName("PRTFILE", serviceProfile + “.prt");

    // Set the perform command to be executed:
    String performCommand = "PERFORM PROCESS WHERE FILEID(“ + serviceProfile + “)”;
    wdiRequest.SetPerformCmd(performCommand);

    // Ask WDI to process the transformation request:
    int rc = wdiTranslator.processRequest(wdiRequest);
  }
  catch (Exception e)   {
    e.printStackTrace();
    throw e;
  }
} // End myTransformService.transform().
```

WDI Service Definition



Service Name (filename): SENDPO

Continue Command Chaining
- [✓ Success]
- Failure
- Always

Description: Command for input XML data

PERFORM Command

```
PERFORM TRANSFORM WHERE SYNTAX(X) INFILE(SENDPO) OUTFILE(OUTFILE) XMLEBCDIC(N) XMLNS(Y) :
PERFORM SEND REQID(GXS_IE) FILEID(OUTFILE):
```

Demo – SOAP Request Message

```
<?xml version="1.0" encoding="UTF-
   8" ?>
- <SOAP-ENV:Envelope
   xmlns:SOAP-
   ENV="http://schemas.xmlsoap.or
   g/soap/envelope/"
   xmlns:q0="http://ibm.com/Transf
   ormService/"
   xmlns:xsd="http://www.w3.org/20
   01/XMLSchema"
   xmlns:xsi="http://www.w3.org/200
   1/XMLSchema-instance">
- <SOAP-ENV:Body>
- <q0:MessageAssembly>
- <Properties>
  <Syntax>xml</Syntax>
  </Properties>
<Message><OrderSR><Header
   typecode="00"><PONum>PO123
   45678901234</PONum><PODate
   >03232001</PODate><Sender><I
   d>OfTheBeast</Id><Qualifier>ST
   </Qualifier></Sender><Receiver
   ><Id>Lewitt</Id><Qualifier>BT</
   Qualifier></Receiver></Header>
   <DetailLoop><ItemNumber>8998
   8760964</ItemNumber><SubDet
   ail><Description>LEG OF
   LAMB</Description><Quantity>1
   .00</Quantity><UnitPrice>5.01</
   UnitPrice></SubDetail></DetailL
   oop><Trailer><ItemCount>6</Ite
   mCount><TotalBucks>1304.55</
   TotalBucks></Trailer></OrderSR
   >
</Message>
  </q0:MessageAssembly>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Demo – SOAP Response Message

```
<soapenv:Envelope
   xmlns:soapenc="http://sch
   emas.xmlsoap.org/soap/e
   ncoding/"
   xmlns:soapenv="http://sch
   emas.xmlsoap.org/soap/e
   nvelope/"
   xmlns:xsd="http://www.w3.
   org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.
   org/2001/XMLSchema-
   instance">
  <soapenv:Header />
- <soapenv:Body>
<p412:TransformMsgResp
   xmlns:p412="http://ibm.com/Tran
   sformService/">ISA*00* *00*
   *ST*OFTHEBEAST *BT*LEWITT
   *060110*1049*U*00401*00000000
   3*0*P*:! GS*PO* *
   *20060110*1049*3*X*004010!
   ST*850*0003!
   BEG*00*NE*PO12345678901234*
   *03232001! N1*ST*OfTheBeast!
   N1*BT*Lewitt!
   PO1*1*****BP*89988760964!
   PO3*ZZ***5.01*FX*1*YY*LEG OF
   LAMB! CTT*6*1304.55!
   SE*8*0003! GE*1*3!
   IEA*1*000000003!</p412:Transfor
   mMsgResp>
  </soapenv:Body>
  </soapenv:Envelope>
```

Summary

- "We have seen the solution and the solution is SOA"
- SOA is the architecture of the future for enterprise applications
  - Advantages of SOA
    - Flexibility - Applications can be rewired and redeployed very quickly
    - Cost effectiveness - Applications can be developed in less time with fewer skills
  - Disadvantages of SOA
    - Difficult to predict loading of shared services like DB (use ORM caching)
    - Limited network bandwidth (Resolve by optimizing for a wire protocol)
- Core components of IBM's SOA vision
  - Service Component Architecture (SCA) Programming Model
  - Service Data Objects (SDO)
  - Common Event Infrastructure (CEI)
- Looked at code showing how to use WDI as a web service in a SOA environment
  - Code is relatively simple and straight forward (if you know what to do)
- Next steps
  - Make your next integration project an SOA project
  - See IBM for more information on WebSphere Process Server and WebSphere Enterprise Service Bus

Questions and Answers

- David Hixon
- IBM B2B Architect
- Tampa, FL
- dhixon@us.ibm.com
- 813-356-5387
- Email me for code samples