Interoperability of Standards
Web Services & .NET, EJB and CORBA

Arne Koschel[1] and Klaus Hofmann zur Linden[1]

[1]IONA Technologies, Brunnenweg 7
64331 Weiterstadt, Germany
{Arne.Koschel, Klaus.Hofmann}@iona.com
http://www.iona.com

Abstract. What is behind "technical interoperability"? It is easy to discuss Web services, but the actual act of integrating services can be extremely complicated. This article will discuss interoperability between Web services, J2EE and CORBA "from PC's over Unix to the Mainframe" highlighting the following technologies: The exposure of Enterprise Java Beans (EJB's) as Web services; The exposure of CORBA Objects as Web services; The interoperability of CORBA and Enterprise Java Beans: CORBA calling EJB's, EJB's calling CORBA.
How do all these technologies communicate? Transactional interoperability becomes increasingly important as more computer languages are developed; security (incl. HTTPS, SAML/XKML, CSIv2, TLS/SSL, RACF) also becomes an issue as the technologies are used to transfer sensitive materials that need to remain private.

1 Introduction

Looking at the discussion about distributed systems recently, it becomes clear, that currently 3 important technologies exist, which are able to support the concept of distributed applications based on broadly accepted standards. Those are Web Services [3,4] & .NET, Enterprise Java Beans/J2EE [2] and CORBA [1]. Those technologies (CORBA, EJB) are already used as the core of 'enterprise level' application systems or are started to be used for B2B-applications across enterprise boundaries or connecting applications of different departments (Web Services & .NET). Such technologies are currently available in various implementations, as usual differing in focus, price and quality, provided by a number of leading vendors like IBM, IONA Technologies, Microsoft and SUN.
Obviously non of those technologies will dominate the software industry on alone, therefore interoperability becomes one of the most crucial issues [7]. The focus of this article will be therefore the technical aspects of interoperability amongst those technologies using a number of code examples. These examples are using IONA's [8] Orbix E2A Application Server Platform (CORBA, J2EE, Web Services) & Orbix E2A XMLBUS as well as Microsoft .NET. Vendor specifics are small and clearly marked in order to demonstrate the principles and to keep them portable.

2 Interoperability Examples

While Enterprise Java Beans and CORBA are already used in many companies as the basis for distributed applications, Web Services are just about to get traction in those environments. Currently a number of large companies are evaluating, if this technology is a viable solution for the integration problem inside and across the firewall.
The basic idea is, that Web Services could introduce a level of abstraction, so that divers applications are made accessible in a uniform way.

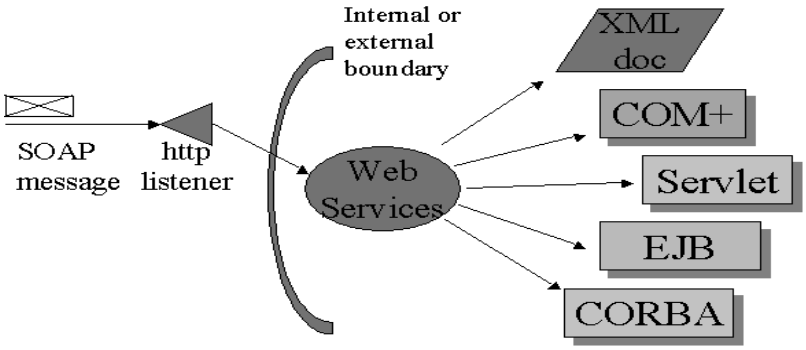


Figure 1 Web Services

Since you usually find a heterogeneous system environment in large enterprises it seems useful, to have a look at the different integration scenarios in detail:
- Possible clients for a Web Service
- An EJB calls a CORBA object
- A CORBA client uses an EJB
- A CORBA Service is provided as a Web Service
- An EJB is published as Web Service

Those scenarios will be in the center of this article. Further aspects of integration as security and transactional behavior will be described as well, but since solutions in this area are only in their very first stages, this will be more of an overview. This includes a brief look at the aspects of process flows for Web Services.

2.1 Possible clients for a Web Service

In order to keep the promise to solve the integration problem for distributed systems in a simple and standards-based way, Web Services need to be accessible by great variety of 'clients' (as a role within an interaction).
Microsoft puts very high emphasis on the Web Services theme with the .NET platform, this makes sure, that a Web Service can be accessed from any .NET based client. To illustrate this, here is how a C# Client invokes a credit card validation service implemented as a Web Service.

```
int num = int.Parse(shortnum);
```

```
int amnt = int.Parse(txt_amount.Text);
...
// How to use the web service in 2 easy steps:
// 1. Connect to the validation service
FNBCreditCardValidationService myClass = new
                              FNBCreditCardValidationService ();
// 2. Call the validateCard method defined on this service
short code =0;
try
{
      code = myClass.validateCard (num, amnt);
}
catch (System.Net.WebException ex)
{
      Console.WriteLine ("Caught Exception" + ex);
      return;
}

      lbl_code.Text = code.ToString();
      cmd_confirm.Enabled = true;
}
```

The FNBCreditCardValidationService.cs contains the following:

```
using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.Web.Services;

System.Web.Services.WebServiceBindingAttribute
                  (Name="FNBCreditCardValidationPortBinding",
Namespace="http://xmlbus.com/FNBCreditCardValidation")]
public class FNBCreditCardValidationService :
System.Web.Services.Protocols.SoapHttpClientProtocol {
    [System.Diagnostics.DebuggerStepThroughAttribute()]
public FNBCreditCardValidationService()
{
      this.Url ="http://klaus:9000/xmlbus/container" +
            "/FNBCreditCardValidation/FNBCreditCardValidatio" +
            "nService/FNBCreditCardValidationPort/";
}

System.Diagnostics.DebuggerStepThroughAttribute();
System.Web.Services.Protocols.SoapRpcMethodAttribute("",
      RequestNamespace = "http://xmlbus.com/"
                              "FNBCreditCardValidation",
      ResponseNamespace = "http://xmlbus.com/" +
                              "FNBCreditCardValidation");
[return: System.Xml.Serialization.SoapElementAttribute("return")]

    public bool confirmPurchase(int param0, System.Single param1, short
param2) {
        object[] results = this.Invoke("confirmPurchase", new object[] {
                    param0,
                    param1,
                    param2});
        return ((bool)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
```

```
    public System.IAsyncResult BeginconfirmPurchase(int param0,
System.Single param1, short param2, System.AsyncCallback callback,
object asyncState) {
        return this.BeginInvoke("confirmPurchase", new object[] {
                    param0,
                    param1,
                    param2}, callback, asyncState);
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public bool EndconfirmPurchase(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((bool)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.Web.Services.Protocols.SoapRpcMethodAttribute("",
RequestNamespace="http://xmlbus.com/FNBCreditCardValidation",
ResponseNamespace="http://xmlbus.com/FNBCreditCardValidation")]
    [return: System.Xml.Serialization.SoapElementAttribute("return")]
    public short validateCard(int param0, System.Single param1) {
        object[] results = this.Invoke("validateCard", new object[] {
                    param0,
                    param1});
        return ((short)(results[0]));
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public System.IAsyncResult BeginvalidateCard(int param0,
System.Single param1, System.AsyncCallback callback, object asyncState)
{
        return this.BeginInvoke("validateCard", new object[] {
                    param0,
                    param1}, callback, asyncState);
    }

    [System.Diagnostics.DebuggerStepThroughAttribute()]
    public short EndvalidateCard(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((short)(results[0]));
    }
}
```

Another important class of clients are Java-based. An integration platform (e.g. IONA's Orbix E2A WebServices Integration Platform XMLBus Edition) generates based on the description of an interface as WSDL (Web Services Definition Language) both a proxy-client for the Java 2 Standard Edition (J2SE) as well as a static client for the Java 2 Micro Edition (J2ME), which can be directly used or which can serve as a template.
Here is, how a Java Client for a Web Service looks like using J2SE:

```
. . .
public float inchToMM(float _param0) throws  RemoteException
{
String soapAction = "";
StringBuffer envelope = new StringBuffer(
      "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
envelope.append("<SOAP-ENV:Envelope ");
envelope.append("\n     xmlns:SOAP-
            ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"");
envelope.append("\n xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"");
envelope.append("\n xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
```

```
                        instance\"");
envelope.append(">\n");
envelope.append("  <SOAP-ENV:Body>\n");
envelope.append("    <m1:inchToMM xmlns:m1=\"urn:target-converter-
                        service\" SOAP-ENV:encodingStyle=
                  \"http://schemas.xmlsoap.org/soap/encoding/\">\n");
envelope.append("      <param0 xsi:type=\"xsd:float\">");
envelope.append(floatToString(_param0));
envelope.append("</param0>\n");
envelope.append("    </m1:inchToMM>\n");
envelope.append("  </SOAP-ENV:Body>\n");
envelope.append("</SOAP-ENV:Envelope>\n");
SOAPResponse_Handler handler =
      new SOAPResponse_Handler("urn:target-converter-service",
                    "inchToMMResponse", "return");
doSoapIO(soapAction,envelope,handler);
if (handler.isFault())
{
      throw new RemoteException(handler.getFaultContent());
}
try
{
    String resultString = handler.getResult();
    if (resultString == null) throw new
      RemoteException("Could not find return in soap response");
      return parseFloat(resultString);
} catch (Exception e)
{
      throw new RemoteException
            ("Error generating result." + e.toString());
}
```

The Method doSoapIO executes the SOAP request:

```
private void doSoapIO(String soapAction, StringBuffer envelope,
                  XMLFilterImpl handler) throws RemoteException
{
. . .
URL url = new URL(soapEndPointURL);
HttpURLConnection connect = (HttpURLConnection)url.openConnection();
connect.setDoOutput(true);
byte bytes[] = envelope.toString().getBytes("UTF-8");

connect.setRequestProperty("SOAPAction","\""+soapAction+"\"");
connect.setRequestProperty("content-type","text/xml");
connect.setRequestProperty("content-length",""+bytes.length);

OutputStream out = connect.getOutputStream();
out.write(bytes);
out.flush();
}
```

2.2 An EJB calls a CORBA object

In order to invoke a CORBA Server out of an EJB, given a J2EE Implementation with the appropriate connectivity, all what is needed is to generate the CORBA ‘stubs’ using the IDL Compiler (which is a part of the CORBA Implementation). If the J2EE Application Server is based on a CORBA Infrastructure it is possible to use the same ORB (Object Request Broker), which manages the EJB. In this case it would be

possible to exchange a transaction context transparently between the EJB Container and the CORBA server, thus enabling a transactional secured communication. This means, that a distributed transaction between the EJB Container and CORBA OTS (Object Transaction Server) can be managed using a 2-phase commit protocol.
The following example is based on the assumption, that the IOR (Interoperable Object Reference) of the CORBA Server is stored within a CORBA Naming Service under 'DirectoryEnquiries' and that this Naming Service is know to the EJB Container. If the IDL of the CORBA Server looks like this:

```
// IDL: EJB_to_CORBA.IDL
module corbaserver {
      struct addressStruct {
            string name;
            string number;
            string street;
            string town;
            string state;
            string zip;
      };
      struct nameStruct {
            string name;
            string number;
      };

      interface ejb_to_corba_int
      {
            void getNumber    (in nameStruct Name,
                               out addressStruct Number);
      };
};
```

The corresponding code inside a method of a Session bean would look like this :

```
// Declaration of CORBA Variables and init
org.omg.CORBA.ORB orb = null;
org.omg.CosNaming.NamingContextExt context = null;
javax.naming.Context jndi_context = null;
org.omg.CORBA.Object objRef = null;
Properties myProps;

// Initialization of the CORBA ORB of the EJB
// The ORB names 'OrbixJ2EE' is preconfigured with some
// properties like security and transaction propagation
Properties orbProperties = new Properties();
orbProperties.put("org.omg.CORBA.ORBClass",
                  "com.iona.corba.art.artimpl.ORBImpl");
orbProperties.put("org.omg.CORBA.ORBSingletonClass",
com.iona.corba.art.artimpl.ORBSingleton");
String[] orbArgs = new String[] {"-ORBname"," OrbixJ2EE"};
orb = org.omg.CORBA.ORB.init(orbArgs, orbProperties);

corbaserver.ejb_to_corba_int directoryEnquiries = null;
NameComponent[] tmpName = new NameComponent[1];
try
{
      // get reference to Naming Service
      objRef = orb.resolve_initial_references("NameService");
      context = NamingContextExtHelper.narrow(objRef);
      // Create Naming Component
      tmpName[0] = new NameComponent("DirectoryEnquiries", "");
      // get object reference
```

```
      objRef = context.resolve(tmpName);
      directoryEnquiries=(corbaserver.ejb_to_corba_int)
                  corbaserver.ejb_to_corba_intHelper.narrow
                  ((org.omg.CORBA.Object)objRef);
}
catch ( org.omg.CORBA.ORBPackage.InvalidName ex )
    …
}

// complete input structure
nameStruct myName = new nameStruct(in_name, "0");
// create output structure
addressStructHolder ns = new addressStructHolder();
// calling the getNumber method
try
{
      System.out.println("BEAN> Calling getNumber()
                  method on DirectoryEnquiries CORBA server object");
      directoryEnquiries.getNumber(myName, ns);
      System.out.println("BEAN> After calling getNumber()");
}
  catch(org.omg.CORBA.SystemException e)
  ...
return ns.value.number;
};
```

2.3 A CORBA client uses an EJB as service

To generate a CORBA interface for a EJB remote-interface ‘rmic’ could be used, but the result is not very ‘nice’ because of the excessive usage of ‘object-by-value’ in the RMI-IIOP Standard. This becomes especially problematic, when a ‘procedural’ programming language is to used for the invoking program. Therefore it seems more practical, if between the CORBA client and the Enterprise Java Bean a Bridge is used, which acts as a CORBA server and encapsulates an EJB client. This CORBA-EJB-Bridge implements a CORBA-IDL, that is designed according to the corresponding Remote Interface and could eventually even be executed ‘in-process’ (depending on the Application Server capabilities).

2.4 An EJB/CORBA service is published as a WebService

To publish an EJB or a CORBA server as WebService a tool is needed, which will generate a WSDL based on the remote interface of the EJB or CORBA IDL and which is capable to receive a SOAP Message and act as a client for the EJB or CORBA server.
As the logic of those 2 cases looks remarkably similar, the following example uses an Enterprise Java Bean. It is a Session Bean with 2 Methods: inchToMM (Inches to millimeter) and mmToInch (millimeter to Inches).

```
...
public class ConverterBean implements SessionBean {

      public float inchToMM(float inches)
      {
            try {
                  return inches * ((Float)new InitialContext().lookup(
```

```
                  "java:comp/env/inchToMillimetreRate")).floatValue();
            } catch (NamingException ex)
            {
                  throw new EJBException(ex);
            }
      }
      public float mmToInch(float millimetres)
                  throws ConversionException {
            try
            {
                  if (millimetres < 0) throw
                        new ConversionException("Test");
                  return millimetres / ((Float)new
                  InitialContext().lookup(
                  "java:comp/env/inchToMillimetreRate")).floatValue();
            } catch (NamingException ex) {
                  throw new EJBException(ex);
            }
      }
. . .
```

The corresponding WSDL looks like this :

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ConverterService"
            targetNamespace="urn:target-converter-service"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:tns="urn:target-converter-service"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsd1="http://xmlbus.com/xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <message name="inchToMM">
            <part name="param0" type="xsd:float"/>
      </message>
      <message name="inchToMMResponse">
            <part name="return" type="xsd:float"/>
      </message>
      <message name="mmToInch">
            <part name="param0" type="xsd:float"/>
      </message>
      <message name="mmToInchResponse">
            <part name="return" type="xsd:float"/>
      </message>
      <portType name="ConverterPortType">
            <operation name="inchToMM">
                  <input message="tns:inchToMM" name="inchToMM"/>
                  <output message="tns:inchToMMResponse"
                              name="inchToMMResponse"/>
            </operation>
            <operation name="mmToInch">
                  <input message="tns:mmToInch" name="mmToInch"/>
                  <output message="tns:mmToInchResponse"
                              name="mmToInchResponse"/>
            </operation>
      </portType>

      <binding name="ConverterPortBinding" type="tns:ConverterPortType">
            <soap:binding style="rpc"
            transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="inchToMM">
                  <soap:operation soapAction="" style="rpc"/>
                  <input name="inchToMM">
```

```
                        <soap:body encodingStyle=
                        "http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:target-converter-service"
                              use="encoded"/>
                  </input>
                  <output name="inchToMMResponse">
                        <soap:body encodingStyle=
                        "http://schemas.xmlsoap.org/soap/encoding/"
                        namespace=
                        "urn:target-converter-service" use="encoded"/>
                  </output>
            </operation>
      </binding>
      <service name="ConverterService">
            <port binding="tns:ConverterPortBinding"
                  name="ConverterPort">
                  <soap:address location="http://localhost:8080/xmlbus/
                  container/Converter/ConverterService/ConverterPort"/>
            </port>
      </service>
</definitions>
```

A tool like IONA's Orbix E2A XMLBus Edition would act as a SOAP-Listener during runtime and behave as a EJB-Client for the Session Bean respectively would play the role of a CORBA-Client for a CORBA-Server.
The architecture of the XMLBus facilitates this taking over the task to make those technologies available as WebServices and to orchestrate those.

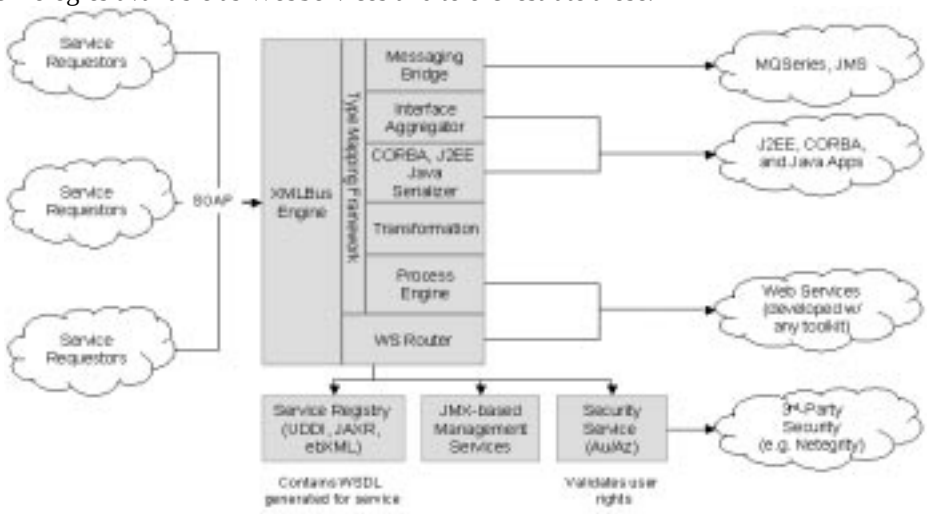


Figure 2 Architecture: Web Services Integration in IONA Orbix E2A

3 "Secure it all"

Security is not to be underestimated for WebServices, as the underlying protocols like tcp/ip or http have to be regarded as insecure. Taking into account that WebServices

will not only be used to integrate different components inside the enterprise but also to enable complex business transactions between 2 ore more business partners via the Internet it becomes very clear, that the requirements for security will be significant. Additionally one has to consider, that WebServices will not be the only technology involved, as the existing technologies will still be used, which already have implemented specific security aspects and which will be confronted with new security requirements being published as a WebService.
Those interactions need to take into account the following aspects :

- Authentication and Authorization services to identify users
- Confidentiality, so that Information is only accessible to authorized participants
- Data integrity, so that information cannot be changed at will or by accident
- Non-repudiation to prove participation in a transaction
- Role based control of access to the different components
- The Possibility to define different security policies
- Flexible Integration with existing security solutions (e.g. Netegrity, LDAP, RACF, Entrust, …)
- The Possibility to encrypt/decrypt in and outgoing messages
- Interoperability through standards
- Mechanisms against Virus or other attacks

The mechanisms to be used to fulfill these requirements and to secure the messages are the following:

- On the communication level the usage of SSL/TLS (Secure Socket Layer / Transport Layer Security) to enable authentication and confidentiality
- On the document level 'XML Encryption' and 'XML Digital Signature' to secure the documents and to enable non-repudiation
- On the recipient level the usage of X.509 certificates

To fulfill these different security requirements especially in the light of the different interoperability scenarios between CORBA, J2EE and WebService applications it needs a security platform, which covers all these different aspects.
As an example IONA's Security Framework (ISF) will be described, which is part of IONA's Orbix E2A products, which was developed with the requirements listed above. The IONA Security Service (IS2) is a central component of the ISF implemented as a Servlet, that can be deployed in IONA's ART (Adaptive Runtime Technology) or a CORBA or J2EE application server like Orbix E2A Application Server, likewise in a number of other application servers like BEA WebLogic , IBM Websphere of Sun iPlanet.

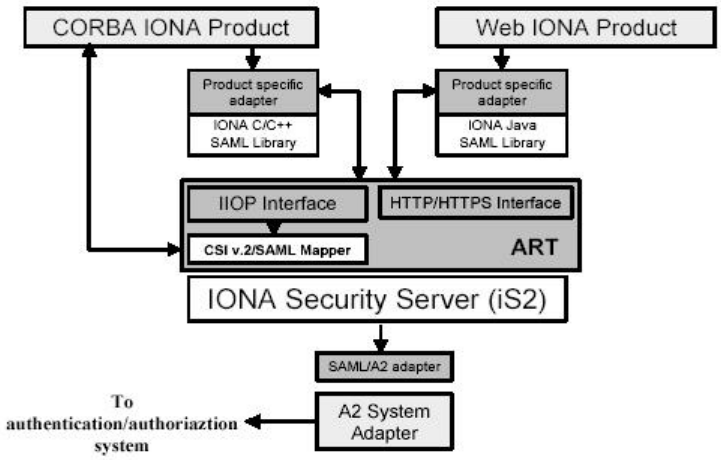


Figure 3 IONA Security Server

Figure 3 shows, how an authorization or authentication request is handled. Applications are using the SAML-functions (Security Assertion Markup Language) to formulate authentication or authorization requests. Web-based applications are communicating with iS2 via HTTP/HTTPS, CORBA or J2EE applications are using IIOP. IS2 offers a number of audit functions and creates log-messages, which makes it easy to integrate with 'enterprise system management' applications like IBM Tivol or HP Open View. IS2 routes the incoming SAML-messages to the authentication and authorization adapter (SAML/A2). This adapter interprets the XML document and passes it to the A2 Adapter, which uses the protocol of the chosen security system to process the request.
On the way back the response gets transformed into the corresponding SAML form and will be returned to the application where the application adapter creates the appropriate 'AuthenticatedPrincipal objects' which include information regarding the identity of the user and his authorities.

4 Process Management for WebServices

Using the WebService technology as the way to integrate various components inside the enterprise and to aggregate those components as business services for the outside world draws the following picture:

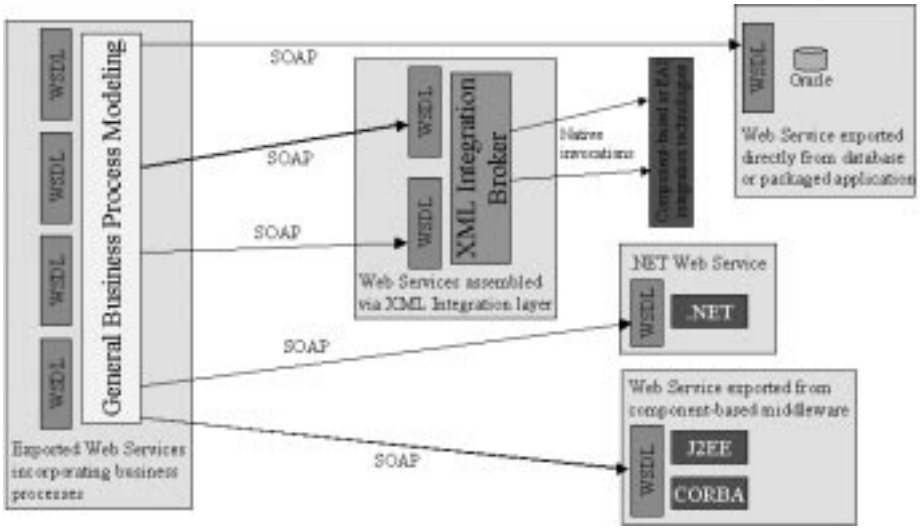


Figure 4 Aggregation of WebServices

Some components export a WebService interface directly, as for an example a database its 'stored procedures' or application functions using a component-based infrastructure like CORBA or J2EE are wrapped as WebService. The figure above also shows the aspects of granularity for WebServices. Not only performance reasons will make it questionable to expose fine-grained WebServices like the attribute of a row in a database table. It makes sense, to integrate applications based on CORBA, .NET EAI and J2EE and to expose aggregated services as WebServices thus facilitating an evolutionary approach.

To the outside world those company internal services will be orchestrated to coarse-grained services offered to business partners via the Intranet or the Internet. In order to implement those business processes a process engine is needed, which manages the different internal services.

Therefore IONA includes a process engine as part of the XMLBus Edition to graphically model and to execute those process chains.

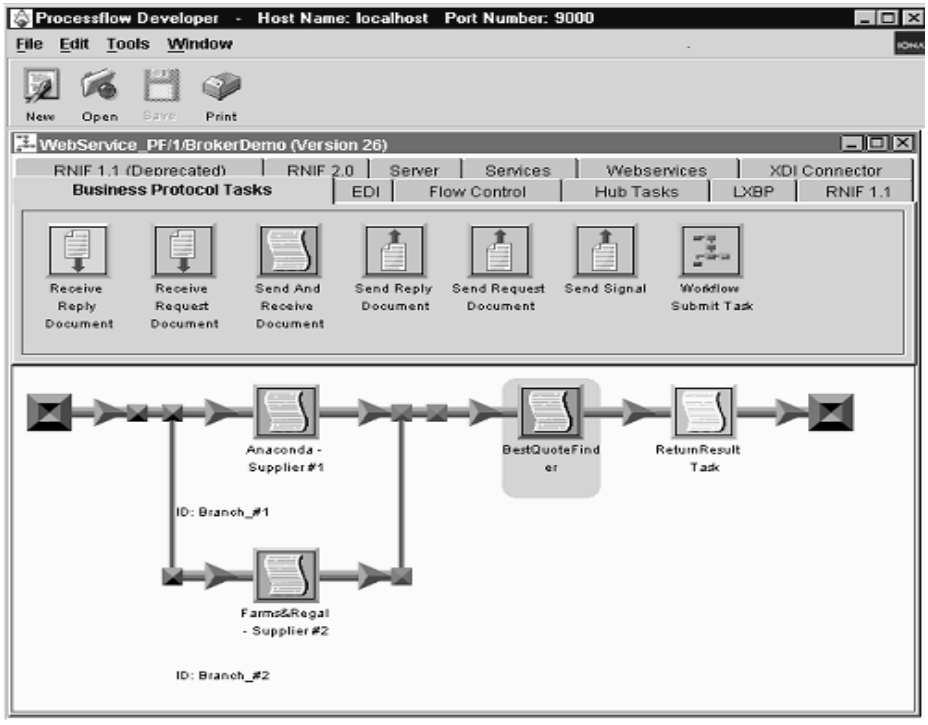


Figure 5 Process Engine

5 Platform examples

A number of vendors are currently integrating WebServices into their environments. According to Gartner vendors who offer a complete e-business platform are e.g. IBM, IONA, Microsoft, Oracle and Sun. As an example a short overview of IONA's platform is given, which offers the necessary features mentioned and which offers WebServices as an integral component of the platform. Integrated means in this context, that Web Service integration is based on a common technology, IONA's Adaptive Runtime Technology (ART). ART is a highly scalable 'plugin framework' to integrate services in a distributed environment based on standards, the groundwork for some of the largest distributed systems. It combines WebServices with aspects like transactions, clustering, security, servlet core, messaging, XML etc. as depicted in Fig. 6

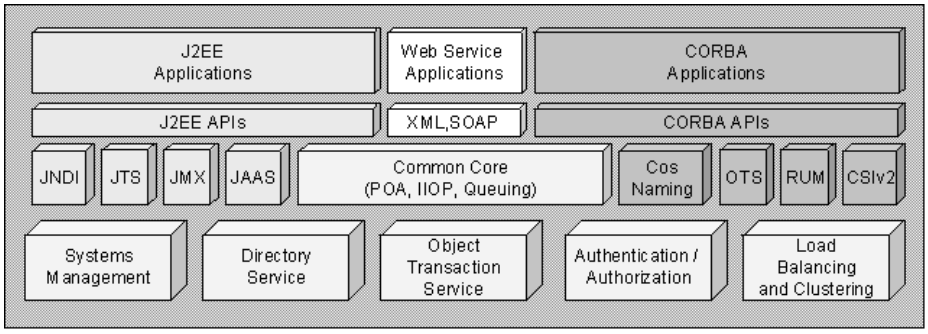


Figure 6 IONA's Adaptive Runtime Technology (ART)

According to Gartner two types of platform will dominate the space of enterprise integration: integration server and application server. Both elements can be found accordingly in the Orbix E2A (End to Anywhere) Platform. The Web Services Integration Platform (Orbix E2A WSIP) contains components for message-based (JMS) application integration (EAI) including adapters and message-transformers as well as components for B2B integration supporting the relevant standards like ebXML or Rosettanet including the appropriate process management, all based on WebServices. The Application Server Platform (Orbix E2A ASP) contains enterprise level CORBA and J2EE components, which could be used separately or in conjunction. WebServices are completing the platform, so that both platforms can work together completely integrated.

6 Conclusion

WebServices are a promising technology that can be seen as the technology groundwork facilitating a service-oriented architecture. Obviously enterprises will still use a variety of different technologies and WebServices can bridge the existing boundaries. Integration has a number of aspects and WebServices can play a crucial role when it comes work with coarse-grained services. As shown in this article one can use different techniques to integrate CORBA and EJB's, which seems to be reasonable, when components need to be integrated so that they can participate in a distributed transaction or to achieve maximized performance or flexibility.
Currently the basis is defined for transaction processing using WebServices, the 'business transaction technical committee' of OASIS (Organization for the Advancement of Structured Information Standards) is working on the specification of BTP (Business Transaction Protocol). As work on 'WS-Transactions' is under way, we will see in the near future, that the software industry will agree upon a number of standards based on WebServices, which will enable the integration of software systems in way, that the usage of the Internet will gain a new quality.

References

1. CORBA, Object Management Group (OMG) www.corba.org
2. J2EE, Sun, java.sun.com/j2ee
3. World Wide Web Consortium, www.w3c.org
4. Web Services, www.webservices.org
5. .NET, Microsoft, www.microsoft.com
6. Hoch-verfuegbare und skalierbare Web/J2EE Appliktionen: Definitionen, Konzpete und Architekturen, A. Koschel, R.Tritsch, ObjektSpektrum 6/2001
7. Web Services zur Integration A. Koschel, R.Tritsch JavaSPETRUM Maerz/April 2002
8. IONA, www.iona.com