Software Reuse and Component-Based SE

ITSE422

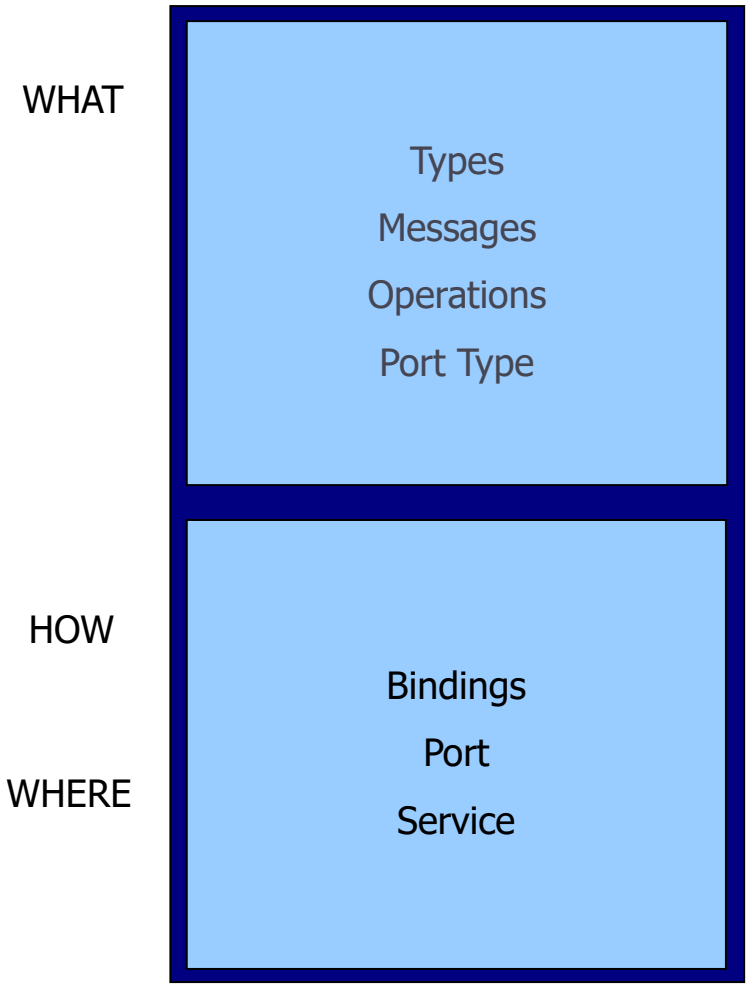
Lecture # 7: Reuse and Composition in Service Computing Part II

Software development with services

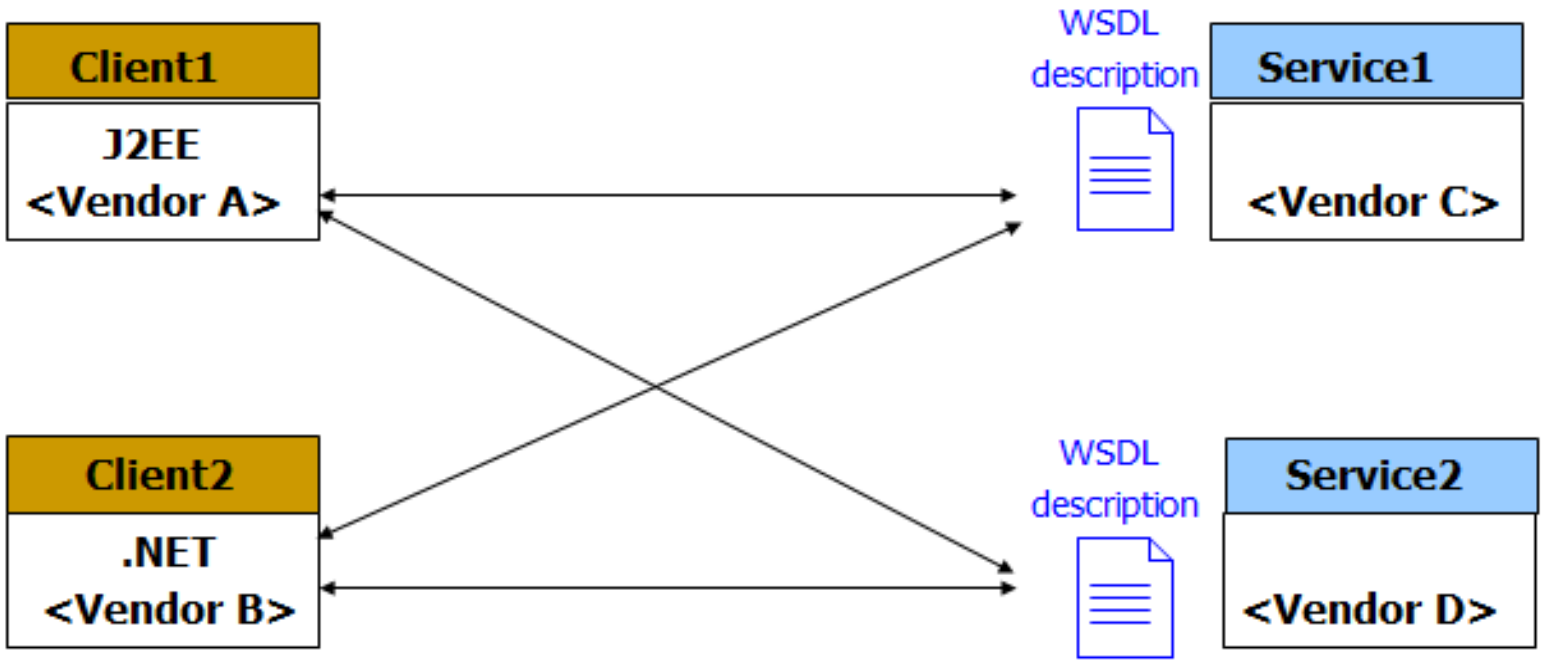
- Simplest case:
  - a client uses ("consumes") a service
- Common case:
  - several services are composed

How Clients Use Services

- In order to use a service, a Client program needs only its WSDL (contains abstract interface description and URI of service endpoint)



Interoperability



How can a client bind to a service ?

- Static binding
  - Service at fixed URL
- Dynamic binding by reference
  - Service URL given at runtime
- Dynamic binding by lookup
  - Look up service URL in registry (need lookup API)
- Dynamic operation selection
  - Service type/operation name given at runtime

- => API's for Web Services: Java, .NET

Java APIs for Web Services

- SOAP messages as Java objects
  - SAAJ (SOAP with Attachments API for Java)
- Programming Model
  - JAX-RPC (Java API for XML-based RPC) => JAX-WS (Java API for XML Web Services)
- Accessing WSDL descriptions
  - JWSDL
- Accessing Web Services Registries
  - JAXR (Java API for XML Registries)

JAX-WS (JAX-RPC)

- WSDL/XML to Java Mapping (wsimport)
- Java to WSDL/XML Mapping (wsgen)
- Client API
  - Classes generated from WSDL
  - Dynamic Proxy
  - DII call Interface

Web Service Example

A Web service AddFunction with operation addInt is known through its WSDL:

```
<wsdl:message name="addIntResponse">
        <wsdl:part name="addIntReturn" type="xsd:int" />
 </wsdl:message>
<wsdl:message name="addIntRequest">
        <wsdl:part name="a" type="xsd:int" />
        <wsdl:part name="b" type="xsd:int" />
 </wsdl:message>
<wsdl:portType name="AddFunction">
     <wsdl:operation name="addInt" parameterOrder="a b">
          <wsdl:input message="impl:addIntRequest" name="addIntRequest" />
          <wsdl:output message="impl:addIntResponse" name="addIntResponse" />
     </wsdl:operation>
 </wsdl:portType>
```

```
// possible implementation of WS:
        // AddFunction.jws
    public class AddFunction {
      int addInt(int a, int b){
          return(a+b);
          }
         }
```

Writing the Client Program

- There are many ways to write a Client program that uses the AddFunction Service (invoking its addInt operation)
  - Using Dynamic Invocation Interface (DII)
  - Using generated Stubs from Service WSDL description
  - Using Dynamic Proxy

Client – using DII

- Using Dynamic Invocation Interface (DII):
  - Service type (WSDL) can be discovered at runtime (WSDL description is actually not even needed !)
  - Service URL is given at runtime (could be extracted from a WSDL)
  - Operation name can also be given at runtime
  - Invocation is done by constructing and sending a call message
  - Most flexible way; but client code looks "ugly"

Client - using DII - Example

```
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.namespace.QName;


    String endpoint = "http://localhost:8080/axis/AddFunction.jws";
    Service  service = new Service();
    Call   call = (Call) service.createCall();
    call.setOperationName(new QName(endpoint, "addInt"));
    call.setTargetEndpointAddress( new java.net.URL(endpoint) );
    Integer ret = (Integer)call.invoke(new Object[]{
                                          new Integer(5), new Integer(6)});
    System.out.println("addInt(5, 6) = " + ret);
```

Client – using generated stubs

- Using generated Stubs from Service WSDL description
  - Service to be used is known from the beginning and the WSDL is available at client development time
  - Service Endpoint Interface (SEI): the (Java) programming language representation of a WSDL port type. Can be generated automatically by tools from a WSDL
  - Stubs (proxies) are classes that implement the SEI. They are generated from the WSDL description (similar with RMI or CORBA middleware for distributed object computing)

Client – using Generated Stubs

Generate the stubs:

**java org.apache.axis.wsdl.WSDL2Java **
http://localhost:8080/axis/AddFunction.jws?wsdl

```
import localhost.*;

    AddFunctionService afs = new AddFunctionServiceLocator();
    AddFunction af = afs.getAddFunction();
    System.out.println("addInt(5, 3) = " + af.addInt(5, 3));
```

Client – using Dynamic Proxy

- Using Dynamic Proxy
  - you need to know the abstract WSDL (port type) at development-time
  - you need to run your WSDL mapping tool against the WSDL document before runtime in order to get the Service Endpoint Interface
  - The proxy (a class implementing the SEI) is obtained at runtime (here is the difference with generated stubs: these are obtained at development time)

Client – using Dynamic Proxy

```
import javax.xml.namespace.QName;
import javax.xml.rpc.*;

        String wsdlUrl = "http://localhost:8080/axis/AddFunction.jws?wsdl";
    String nameSpaceUri = "http://localhost:8080/axis/AddFunction.jws";
    String serviceName = "AddFunctionService";
    String portName = "AddFunction";
    ServiceFactory serviceFactory = ServiceFactory.newInstance();
    Service afs = serviceFactory.createService(new java.net.URL(wsdlUrl),
            new QName(nameSpaceUri, serviceName));
    AddFunctionServiceIntf afsIntf = (AddFunctionServiceIntf)afs.getPort(
            new QName(nameSpaceUri, portName),AddFunctionServiceIntf.class);
    System.out.println("addInt(5, 3) = " + afsIntf.addInt(5, 3));
```

Where and How to find Services ?

- Service registries:
  - UDDI
    - Standard for representing and organizing registry information
    - Standard API's for:
      - publishing services on UDDI registry
      - Lookup services from registry
    - Private UDDI registries: inside one enterprise
    - Public UDDI registries:
      - Existed maintained by major companies
      - Not anymore (since 2008)
    - Problems of UDDI: (why public UDDI registries died):
      - Complex standard and API
      - No semantic information
      - No certification, no trust
      - Info published in UDDI registry accessible only via UDDI lookup API's, not accessible via usual search engines
- Using usual search engines
- Using Web service search engines
  - http://webservices.seekda.com/

- Questions?