Vrije Universiteit Brussel

The Service Composition Environment

Niels Joncheere
System and Software Engineering Lab (SSEL)
Vrije Universiteit Brussel

Web services

→ exposing heterogeneous applications to a network using XML-based standards for interface description (WSDL), messaging (SOAP), and discovery (UDDI)

A short history of web service composition

General-purpose programming languages

Java

```
package com.ibm.wsdk.Sample2;

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

public class Dii {

  public static void main(String[] args) {

    String targetNamespace = "http://Sample2.wsdk.ibm.com";

    try {
      /* Service lookup */
      ServiceFactory serviceFactory = ServiceFactory.newInstance();
      javax.xml.rpc.Service service = serviceFactory.createService(
          new QName(targetNamespace, "HelloInterfaceService"));

      /* Service access */
      Call call = (Call) service.createCall();
      call.setProperty(Call.ENCODINGSTYLE_URI_PROPERTY, "");
      call.setProperty(Call.OPERATION_STYLE_PROPERTY, "wrapped");
      call.setTargetEndpointAddress(
          "http://localhost:8080/Sample2WebService/services/Sample2");
      call.removeAllParameters();
      call.setPortTypeName(new QName(targetNamespace, "Sample2"));
      call.setOperationName(new QName(targetNamespace, "getGreeting"));
      if (call.isParameterAndReturnSpecRequired(call.getOperationName())) {
        call.addParameter(
            "arg0",
            new QName("http://www.w3.org/2001/XMLSchema", "string"),
            String.class,
            javax.xml.rpc.ParameterMode.IN);
        call.setReturnType(
            new QName("http://www.w3.org/2001/XMLSchema", "string"));
      }

      /* Service invocation */
      System.out.println(call.invoke(new Object[] { "Jane" }));
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Dedicated workflow languages

BPEL

```
<?xml version="1.0" encoding="UTF-8" ?>
<process name="Demonstrator"
    targetNamespace="urn:tlcpf.alcatel.be"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process"
    xmlns:tns="urn:tlcpf.alcatel.be"
    xmlns:wiw="urn:spf.alcatel.be/WhoIsWho">

  <partnerLinks>
    <partnerLink name="whoiswhoPL"
        partnerLinkType="tns:whoiswhoPLT"
        partnerRole="whoiswho" />
  </partnerLinks>

  <variables xmlns:xsi="http://www.w3.org/2001/XMLSchema">
    <variable name="whoiswhoRequest" messageType="wiw:getEmployeeByInternalPhone1In" />
    <variable name="whoiswhoResponse" messageType="wiw:getEmployeeByInternalPhone1Out" />
  </variables>

  <sequence name="mainSequence">
      <invoke name="whoiswhoInvoke" partnerLink="whoiswhoPL"
          portType="wiw:WhoIsWho" operation="getEmployeeByInternalPhone"
          inputVariable="whoiswhoRequest" outputVariable="whoiswhoResponse" />
  </sequence>
</process>
```

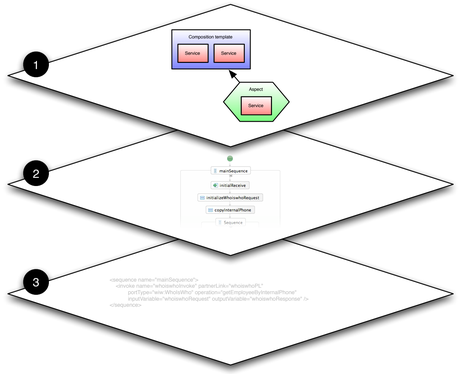
???

→ composition logic is nicely modularized (+)
→ programming at low abstraction level (-)

→ composition logic is lost (-)
→ programming at very low abstraction level (-)

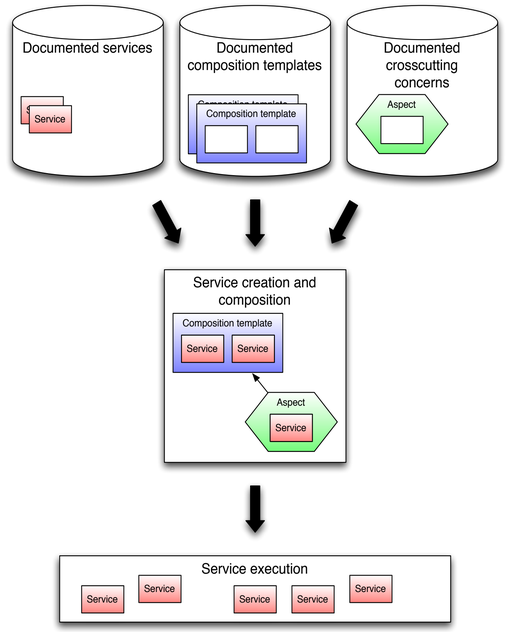
Approach

→ several layers of abstraction:



→ web service composition at a higher level of abstraction, as plug-and-play composition of existing building blocks (for domain experts) (1)
→ visual editing of BPEL processes (for composition experts) (2)
→ textual editing of BPEL processes (for composition experts) (3)
→ easy switching between abstraction levels
→ guiding developers in creating valid compositions

The Service Composition Environment (SCE)



Prototype highlights

→ supports general-purpose AOP using Padus (AOP language for BPEL):

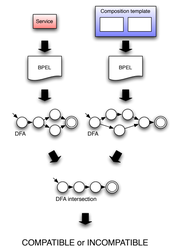
Padus

```
<aspect ...>
  <using>
    <namespace ... />
    <partnerLink ... />
    <variable ... />
  </using>
  <before joinpoint="Jp"
          pointcut="invoking(Jp, 'MyService', 'MyPortType', Op)">
    <bpws:sequence>
      <bpws:assign>...</bpws:assign>
      <bpws:invoke ... />
    </bpws:sequence>
  </before>
</aspect>
```

→ introduction of namespaces, partner links, and variables
→ pointcuts
→ advices

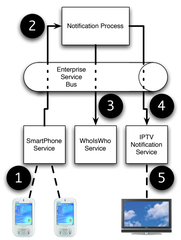
→ supports concern-specific languages for common concerns such as billing

→ verifies compatibility of services with composition templates (on interface and protocol level)



→ performs basic quality-of-service checking

Use case



Future work

→ pointcut visualization
→ framework for concern-specific languages
→ framework for quality-of-service properties

Niels Joncheere
System and Software Engineering Lab (SSEL)
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
njonchee@vub.ac.be
http://ssel.vub.ac.be/ssel/about:members:nielsjoncheere