Vet-Trend - Transnational Meeting
Integrating virtual and remote laboratories using Web-Services

Dipl.-Inf. Andreas Rasche
Dr. Leandro Soares Indrusiak
Outline

- The Vision
- XML-based Web-Services
  - Overview
  - Programming Models
- The Adaptive (Web Service) Execution Plattform
  - Stateful Web-Services
  - On-demand Web Service Deployment
- Integrating the laboratories
  - Case study Potsdam -- Darmstadt
  - Batch-Mode vs. Interactive-Mode
The Vision

Adaptive Execution Platform (HPI Potsdam)

Execution Host (TU Darmstadt)

Fourcault's Pendulum

Execution Host (HPI Potsdam)

Execution Host (Uni Transilvania Brasov)

Experiment B

Experiment A

Higher Slikker

LEGO® NXT

Student (Uni Genova)

Student (NU Lissabon)

Student (ÖTH Ronneby)
Web Services Overview

- XML-based middleware for communication
- Transport typically over HTTP and TCP/IP
- Several extensions
  - WS-ResourceProperties
  - WS-ResourceLifetime
  - WS-Addressing (EndpointReference)
  - WS-Security
- In practice many interoperability problems
  - Different encoding styles for method parameters in WSDL
  - Complex data types (HashMap vs. HashTable)
  - Document/literal encoding (WSI) works for Java and .Net
    - .NET: .NET 2.0 WSE 3.0
    - Java: JAX-WS 2.1, Java 6.0
The Adaptive Execution Platform

- Infrastructure for dynamic deployment and execution of web services
- Logical services
  - Represent end-point for a service request
  - 1:n mapping to physical service instances
  - Priority-based scheduling of invocations
  - Support of service invocation cancelation
- Physical services
  - Physical deployed service instance
- Service Properties using WS-ResourceProperties
  - Global properties (available to all instances)
  - Instance properties (unique per instance)
AXP Architecture
Batch mode processing
Each experiment is represented by a service
  Independent compile services for translating experiment control code
Each experiment usage is a service invocation
  Service method invocation returns after completion of the experiment run
Experiment results can be accessed via properties
Service deployment descriptor contains experiment information
AXP Lab Architecture
Implementing an Experiment Web Service

```csharp
[WebService(Namespace="http://hpi-web.de/ExperimentServices")]
public class NxtExecutionService : WebService
{
    [WebMethod]
    public void ExecuteExperiment(byte[] experimentData)
    {
        ... Load data to expriment and execute
        // save experiment result
        PropertySupport.SetInstanceProperty("LegoPath", path);
        PropertySupport.SetInstanceProperty("StateFlow", flow);
    }
    [WebMethod]
    public void Cancel(bool isUserCancelling)
    {
        // handle cancelation
    }
}
```
<?xml version="1.0" ?>
<deployment Assembly="ExecutionService.dll">
  <platform>dotnet</platform>
  - <platformRequirements>
    <name>Bluetooth</name>
  </platformRequirements>
  - <platformRequirements>
    <name>LegoNXT</name>
  </platformRequirements>
  - <service>
    <serviceID>q1:NxtExecutionService</serviceID>
    <description>NxtExecutionService</description>
    <schedulingMode>serialized</schedulingMode>
    <executionDelay>3</executionDelay>
    - <property>
      <name>ExperimentType</name>
      <defaultValue>LegoNxt</defaultValue>
      <scope>service</scope>
    </property>
    - <property>
      <name>RenderTypes</name>
      <defaultValue>Result (Text)</defaultValue>
      <scope>service</scope>
    </property>
    - <property>
      <name>SampleCode</name>
      <defaultValue>class Controller{public void Drive(){Robot.Go();}}</defaultValue>
      <scope>service</scope>
    </property>
    - <property>
      <name>Experiment Description</name>
      <defaultValue>Control a Lego NXT Robot</defaultValue>
      <scope>service</scope>
    </property>
    - <property>
      <name>Learning Objectives</name>
      <defaultValue>Robot Control;Real-Time Systems</defaultValue>
      <scope>service</scope>
    </property>
  </service>
</deployment>
A Web Service Client

ServiceInfo[] services = deploymentService.ListServices();

... Select experiment service ...

ServiceFactoryService serviceFactory = new ServiceFactoryService(factoryIp);

EndpointReference expService = serviceFactory.CreateServiceInstance(serviceId, priority, false);

UsernameToken secToken = new UsernameToken(parameters.User, parameters.Password, PasswordOption.SendHashed);
expService.RequestSoapContext.Security.Tokens.Add(secToken);

expService.ExecuteExperiment(controlFile.ReadToEnd());

WsrPropertiesService properties = new WsrPropertiesService(expService);
properties.GetResourceProperty<byte[]>("LegoPath");
Case Study: Potsdam -- Darmstadt

- TU Darmstadt's remote prototyping lab
  - three releases so far:
    - batch mode based on sockets
    - batch mode based on Jini
    - interactive mode based on Jini and Ptolemy II
Case Study: Potsdam -- Darmstadt

client

reconfigurable hardware module

reconfigurable hardware service & lookup service
Case Study: Potsdam -- Darmstadt

client: Ptolemy II

queries for prototyping platforms

receives proxy of remote resource

Jini registry and lookup server

JavaSpaces Ticketing Service

upload service proxy and ticket

backend

backend

backend

pool of FPGA-based prototyping platforms
Case Study: Potsdam -- Darmstadt
Case Study: Potsdam -- Darmstadt

- Goal: integrate TU Darmstadt's remote prototyping lab to Potsdam AXP Lab
  - substitute Jini for webservice
- Synergy: take advantage of AXP Lab
  - use available front-ends: simpler and standard
  - use available service handling mechanisms
  - potential to employ simplified authentication mechanism
- Plus:
  - another case study to AXP Lab
  - driver for AXP improvements: support interactive mode
Support for Interactive Modes

- In batch mode only data transfer at begin and end of experiment usage possible
- Interactive mode requires data transfer from/to user during experiment usage
- Solution: Usage of additional stream-oriented Web Service methods
  - byte[] ReadData()
  - WriteData(byte[] inData)
- The infrastructure must be extended to support concurrent request on a physical service instance
Conclusions

- Batch mode remote experiments can be interconnected using Web Service
  - Interoperability between platforms, operating systems, languages supported
  - Firewall problems solved
  - Security support
- Connection between HPI Potsdam and TU Darmstadt will be running soon
- Open questions
  - Integration of other experiment types
  - Experiment service repository and description