Efficient Data Transmission Between Multimedia Web Services via Aspect-Oriented Programming

Dominik Seiler\textsuperscript{1,2}, Ernst Juhnke\textsuperscript{2}, Ralph Ewerth\textsuperscript{2}, Manfred Grauer\textsuperscript{1}, Bernd Freisleben\textsuperscript{2}

\textsuperscript{1} Information Systems Institute, University of Siegen, Germany
\textsuperscript{2} Dept. of Math. and Comp. Science, University of Marburg, Germany
Motivation

• Plethora of multimedia web services in WWW
  – Image and video data sources
    • Flickr, YouTube etc.
  – Data processing
    • Image/video OCR, transcoding, face detection…

• Goal: Create new applications by service composition
  – Compose a value-added workflow
  – Easier development & integration
Problem Statement

• Assume you want to have a workflow of multimedia web services with
  – secure data transmission, and/or
  – reliable data transmission, and/or
  – metadata management, and/or
  – workflow modeling support (BPEL), and other features
• SOAP services offer tool support for such requirements
  – in contrast to RESTful web services
  – but multimedia web services potentially deal with large data
• Problem
  – Data transmission between SOAP web services
  – Workflow engine can become a bottleneck
Our Approach

• Consider data transmission as a cross-cutting concern in workflows
  – Data handling has to be addressed in many/all components
  – Address this issue by *Aspect-Oriented Programming*
• Reference technique optimizes data transmission

• Solution
  Aspect-oriented framework for efficient data transmission
Introduction – What is *Aspect-Oriented Programming*?

- Aspect-oriented programming (AOP)
  - Aims at increasing modularity of software systems
  - Encapsulates cross-cutting concerns into advice
- Integration into existing applications via join points
- An aspect combines
  - Point cuts: description of a set of join points
  - Advice: code to be executed at specific join points
Intro – BPEL Workflows

• Business Process Execution Language (BPEL)
• Standard for service composition
• General purpose workflow language
  – Turing-complete
  – Exposed as a web service
  – Basic/structured Activities
• Explicit modeling of control flow
• Excellent tool support
Introduction – Flex-SwA

• Flexible handling of bulk data
  – Service-oriented environment
• Reference builder
  – Creates XML description
• Reference handling
  – Transparent for the workflow provider
• Avoid bottlenecks due to file transfers
Request/Response Aspects

• Adapting data transmission requires modifications
  – On both, client and server side
  – Code must be aware of modifications
  – Adaption across different administrative domains
• Solution: Weave request/response aspects at message level
• Examples for non-functional requirements in web services
  – Data transmission
  – Security
  – Reliable messaging
Design

- Remain independent of web service implementation
- Client-side at BPEL engine (or any other client)
- Server-side at application server (heterogeneous administration domains)
Design – Client-side

- Invoke activities are annotated to use the `Aspect-InvokeHandler` (AIH)
- AIH weaves request/response-aspects into services
  - Ensures atomic behavior
- Unchanged implementation
- Transparent for the workflow
Design – Server-side

• Aspect Configurator
  – Add, remove, check
• Security Manager
  – PKI-based
• AspectProvider
  – Weaving component
  – Based on AspectJ
Implementation (server-side)

- AspectProvider
  - Woven into the Axis handler chain
  - Applies request/response on SOAP messages
Implementation (client-side)

- Schema-type of a request/response aspect

```xml
<complexType name="Aspect">
  <sequence>
    <element name="portType" type="xsd:QName" />
    <element name="operationName" type="xsd:string" />
    <element name="field" type="xsd:string" />
    <element name="mode" type="xsd:string" />
    <element name="aspectPlugIn" type="xsd:string" />
    <element name="aspectData" type="tns1:HashMap" minOccurs="0" />
  </sequence>
</complexType>
```
Experimental Setup – Testbed

• Web services
  – MPEG decoder, face detector, MPEG-7 converter

• Web service environment
  – Tomcat 6, Axis 1.4, ActiveBPEL

• Computational environment: Amazon EC2
  – High-CPU Medium Instances
Experimental Setup – Test Scenarios

Test scenario I
- Plain SOAP
- 2 EC2 machines
  - BPEL engine
  - Service container

Test scenario II
- AOP / Flex-SwA-Aspect
- 3 EC2 machines
  - BPEL engine
  - MPEG decoder
  - Face detector, MPEG-7 converter
Experimental Setup – Request/Response Aspects

Aspect serviceAspect = new Aspect(
    new QName(
        "http://fb12.de/MpegDecoderService",
        "MpegDecoder"),
    "getNextFrame",
    "/0/imageData",
    Aspect.AOP_RESPONSE_MODE,
    "FlexSwAPlugIn")

- void setVideo()
- DataBean getMetaData(...)
- ImageBean getNextFrame(...)
- ImageBean getFrameNumber(...)

ImageData
  + frameNumber
  + imageData
  + ...
Experimental Setup – Results

• Comparison of the two test scenarios
• Impact
  – Large improvement
  – Negligible overhead
Conclusion

• Aspect-oriented framework for SOAP multimedia web services
  – Message based
  – Efficient data transmission between web services
• Reference-based multimedia data transmission
• Reduced development efforts
  – Benefit easily from rich tool support of SOAP web services
• Future work: integration of more sophisticated AOP mechanisms
Thank you for your attention!

Any questions or remarks