Integrating WS-Agreement with a Framework for Service-Oriented Infrastructures

Run WS-Agreement services within the SLA@SOI Framework

Date: 26/10/2010
Speaker: Peter Chronz, Philipp Wieder
Organisation: TU Dortmund, Service Computing Group
Outline

What offers SLA@SOI?
Why syntax conversion?
Methodological approach: The SLA* model
Technical approach: A generic SLA manager
Conversion example
Open issues
• **SLA management framework**
  ◊ Harmonizing perspectives of relevant stakeholders (software/service/infrastructure provider and customer)
  ◊ General methods for SLA specification and negotiation & systematic multi-layer SLA management (planning, optimization, and provisioning), monitoring and accounting

• **Adaptive SLA-aware infrastructures**
  ◊ Well defined interfaces for adaptive infrastructures with harmonized access to different virtualization technologies.
  ◊ Advanced technologies for SLA enforcement on infrastructure level

• **Engineering methods for predictable service-oriented systems**
  ◊ Modelling techniques and prediction tools for SOA and SOI components

• **Business management suite for e-contracting**
  ◊ Covers complete business lifecycle of a service provisioning and delivery
An Application Scenario

![Diagram of an application scenario with layers and interactions between components such as GDI-SLA Broker, SLA4D-Grid service layer, Infrastructure SLAM, VM, WPS, and Cloud Interface (Tashi, OpenNebula, ...).]
Motivation for Syntax-Conversion

Various types of renderings
- SLA@SOI’s SLA model allows various types of renderings
  - XML
  - RDF/OWL
  - JSON
  - ...
- Framework internal representation is Java
- Essential: Conversion between string-based representations and Java-based representations

Various types of SLA models
- SLA@SOI supports various types of SLAs
  - Its own SLA model
  - WS-Agreement with embedded content
  - Potentially further models
- Framework internal model is SLA*
- required: Conversion between different SLA models
The Basis: SLA@SOI’s SLA Model

Provides various representations
- XML, JSON, Java
- ...

Uses vocabularies as extensions to the core model
- Business Terms
- Infrastructure Terms
- QoS Terms

Validates content automatically
- Makes use of external vocabulary
- Validation included e.g. is the data-type right? Is the value in the predefined range?

Has been developed based on WS-Agreement extensions
- Domain-specific extensions for certain domains targeting e.g. the infrastructure layer
Syntax-Conversion in Action
Approach

Ideal versus pragmatic

• Pragmatic approach has been chosen to ease the implementation

• Potential problems
  ◦ Semantics of mapping may not be clear directly
  ◦ Documentation essential for programmers/adopters

Bi-lateral mapping necessary

• Realise domain-specific WS-Agreement extensions as SLA* elements

SLA* provides in general simple solutions

• E.g. XPath for XML-based systems versus SLA* PATH
Example Mapping

**WS-Agreement ↔ SLA***

- **Top-level**
  - Name ↔ SLA identifier

- **wsag:GuaranteeTerm to slasoi:Guaranteed**
  - Obligated ↔ Precondition of a guaranteed state or action
  - QualifyingCondition ↔ Also goes into the precondition of state/action
  - ServiceLevelObjective ↔ Guaranteed state
  - BusinessValueList ↔ No explicit support in SLA@SOI

- **BusinessValueList to SLA***
  - Penaltys/Rewards ↔ E.g. custom guaranteed action with AssessmentInterval and Count as precondition
Selected Conversion Issues

Constraint expressions
- WS-Agreement uses XML-Schema-based constraints
- SLA@SOI's model has its own constraint concept
- WS-Agreement constraints cannot be mapped to SLA* constraints in all cases
- Using SLA@SOI as free-formed constraints in WS-Agreement is one solution (but then standard validation does not work)

Business terms
- Only supported by SLA@SOI via an external vocabulary
- Not included in core model

Many minor issues
- Missing elements and attributes in SLA@SOI need to be supported by complementary vocabularies
- Such extensions are foreseen in SLA@SOI
Thank you!
# Colour Scheme

## Main colours (according to branding scheme)

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>0</td>
<td>153</td>
<td>255</td>
</tr>
<tr>
<td>103</td>
<td>204</td>
<td>255</td>
</tr>
<tr>
<td>221</td>
<td>221</td>
<td>221</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
</tbody>
</table>

## Additional colours (to be used for fancy diagrams)

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>204</td>
<td>255</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>153</td>
</tr>
<tr>
<td>204</td>
<td>153</td>
<td>102</td>
</tr>
<tr>
<td>204</td>
<td>255</td>
<td>204</td>
</tr>
</tbody>
</table>