SLA and SLA Template Model

Keven Kearney
Organisation: Engineering Ingegneria Informatica, Italy
Date: 28/09/2010
Overview

Model for Service Level Agreements (SLAs) and SLA Templates.

3 Key Objectives:

- support domain independent solutions at all stages of SLA(T) life-cycle,
- permit domain-specific extensions / customisation,
- language & technology independence
Objective 1: Domain Independence

Managed SLA(T) Lifecycle

Design
→
(QoS-Based) Discovery
→
Negotiation
→
Provisioning
→
Execution & Monitoring

Characterised as ...

Constraint Satisfaction Problems
→
Fine-grained specification of “constraints”

(QoS = Quality of Service)
Objective 2 : Domain Specificity

Diverse Use-Cases
- ERP Hosting
- Enterprise IT
- Live-Media Streaming
- Health-Care Provision

Diversity of ..
- Service Types (web-based v human-based)
- QoS Parameters
- Business Models (pricing / penalties)

Extensible & Customisable Service & QoS Terms
An SLA is an SLA no matter what language it is written in ...

- XML (cf. WS-Agreement)
- OWL (cf. semantic-based service descriptions)
- Human-readable formats

- **WS-Agreement:**
  - “An SLA is an **XML** document with properties x, y, z, …”

- **SLA@SOI:**
  - “An SLA is **any** document with properties x, y, z, …”

"Abstract" specification of SLA(T) Content
<table>
<thead>
<tr>
<th>Objectives</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fine-grained specification of constraints</strong></td>
<td>• The core of the model is a well-defined constraint language.</td>
</tr>
<tr>
<td><strong>Extensible &amp; Customisable</strong></td>
<td>• Pluggable domain-specific “vocabularies”</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
<td>• The model specifies an “abstract syntax”</td>
</tr>
<tr>
<td></td>
<td>• Concrete Implementations</td>
</tr>
</tbody>
</table>
Abstract Syntax ?

Abstract Specification ...

- Function *subtype-of* Value
- *Definition*: an ordered pair \(<f, P>\) where:
  - \(f \in UUID\) identifies the function,
  - \(P\) is an ordered set \(\{p_1, p_2, \ldots, p_n\}\) where \(p_{0<i<n} \in Param\)

Concrete Examples ...

- “the completion-time of X” (human-readable)
- “completion_time( X )” (BNF)
- “<function uuid=’completion-time’>
  <param>X</param>
</function>” (XML)
“Service levels” are modelled as constraints ...

- “completion-time( X ) < 10 s”
- all aspects of this expression can be customised

Built-in support for ‘constraint satisfaction’
e.g. service discovery ...

- Provider offers: completion-time( X ) <= 1000 ms
- Customer query: completion-time( X ) < 1 s
- Problem: given the offer, can we guarantee that the customer’s requirements will be satisfied ?
- To answer:
  ◊ data conversion formula – e.g. “X s” = “X*1000 ms”
  ◊ semantic relations between comparison operators – e.g. that ‘<’ is more stringent than ‘<=’
Vocabularies: Modular Design

- **CORE**
  - Abstract constraint language (incl. standard comparison/logic/arithmetic operators)
  - Vocabulary Specification

- **UNITS**
  - Common metric unit definitions (incl. unit conversion formula)

- **SERVICE INTERFACE**
  - Abstraction of WSDL 2.0

- **SLA(T)**
  - SLA & SLA Template Content (parties, service level objectives, guarantees)

- **QoS**
  - Common QoS Parameters

- **Business**
  - Pricing & penalty information, termination clauses
Validation Example

QoS Vocabulary definition of “completion-time”

Use of “completion-time” in an SLA Template
Validation Example

File: /Users/KevenT/Documents/workspace/_STAR/test_resources/negative/slat014b.xml
Validating ..
Importing "http://www.slaatsoi.eu/star-core"
Importing "http://www.slaatsoi.eu/star-sla"
Importing "http://www.slaatsoi.eu/star-iface"
Importing "http://www.slaatsoi.eu/star-units"
Importing "http://www.slaatsoi.eu/star-qos"

---------
ERROR [row:53 col:49] (58) Cardinality error on parameter at index 0: found array with 2 item(s) (cardinality is specified as '1')
---------
ERROR [row:53 col:49] (67) Unable to resolve result type of Function 'qos:completionTime'
---------
Validation failed
---------
Found 2 errors and 0 warnings

(time taken = 155 ms)
Summary

Domain-independence
• ‘constraint satisfaction’ is an issue at all stages of SLA(T) life-cycle (discovery, negotiation, monitoring)
• Core the model is a well-defined & fine-grained constraint language
• Modular & minimal specification (use only what you need).

Extensibility / Customisability
• Supports modular extension through domain-specific vocabularies
• Default vocabularies for common operators, metric units, QoS & business terms

Language (or Technology) Independence
• Model is ‘abstract’ : defines the form, but not the tokens
• Generic notion of Service & SLA (not restricted to Web-Services)
• Compatible with existing XML standards (WSDL & WS-Agreement), but not restricted to XML.