A White Paper on a Service Validation language

COMS 4115 Project
Shrirang Gadgil [CVN]

Introduction

During past few years, telecom service provides are moving from pure traffic bearer services like VPNs to higher level of services. With glut of bandwidth available in core networks, distributed and complex services can be easily built. On computing front, companies like IBM and HP are trying to promote on-demand computing services. Services are currently defined using XML where resource requirements, service dependencies and service guarantees are defined. Such complex definitions have to be validated using different kinds of rules in different sectors, such as, technology, vendor idiosyncrasies or purely business policies. Validation policies are also desirable to control behavior of validation (for example the strictness of validation rules)

Goals

Goals of this project are following

1. Design a language for defining service validation rules in very simple way
2. Implement a compiler for this language
   The rules written in this language are compiled in to Java classes than can be invoked for validation of services defined in XML
3. Invocation of the validation classes provides results of validation process

Non-goals

It is assumed that services are defined in XML in a hierarchical fashion. It is not the goal of this project to define a language for service definition. The author of the rules is assumed to aware of service definition formats. It is not a goal of the project to automatically understand service definitions and check the propriety of rules during compilation.

Sample syntax

Although rigorous syntax has not been worked out completely, this section shows a sample of rules that are being envisaged. While designing the language, we want to take maximum advantage of the fact that these rules will be translated into Java classes.

A sample rule source code is as follows

File: Sample.rules

```
package com.mypackage;
import com.otherpackage.Sample0;

private rule A validates subelementtype subelement
{
   condition c1 (subelement.attribute0 == somevalue);
   condition c2 (subelement.attribute1 == somevalue);

   result
   {
       c1 && c2;
   }
   logonsuccess (“Rule A passed”);
   logonfailure (“Rule A failed”);

}

public rule main validates myelementtype myelement
{
   subrule A a processes myelementtype.subelement if
   {
      condition (myelement.attribute1 == somevalue);
   }
   subrule Sample0.B b processes
                          myelementtype.subelement if
   {
      subrule Sample0.C (myelement);
   }

   result
   {
      a & b;
   }
   logonsuccess (“Rule main passed.”);
   logonfailure (“Rule main failed.”);
   logonnouse (“Rule main not triggered.”);

}
```

Note: All the words in blue are keywords.

The rules written above intend to validate a xml service definition which is assumed to be a hierarchical structure of elements and their sub-elements with their attribute values.

The main rule contains two constituent two subrules A and B that are invoked conditionally. If conditions are not met the main rule returns don't care. In case of AND operation don't cares are evaluated as 1's and in OR operations don't cares are evaluated as 0's. The result construct contains a Boolean expression for evaluation of the rule in terms of results of subrules A and B.

The subrule A is programmed as comprising of two constituent conditions. The result construct has a boolean expression of the constituents.