Symbolic Evaluator Service

O. Caprotti*
RISC-Linz
Johannes Kepler University
Linz, Austria
olga.caprotti@risc.uni-linz.ac.at

May 8, 2002

Abstract

In this report we describe the architecture and the installation of a symbolic evaluator service. This work is part of milestone 1.0 for the MathBroker project and represent a case study of available software for deploying services via a web server that use WSDL as interface.

1 Introduction

This report is a detailed description of the development and installation of a mathematical web service based on available software that provides WSDL interface and the actual interface to the mathematical computational engine.

The goal of this document is to report our experience, to explain the components of the archicture, and to provide guidelines that allow the reader to install his/her own mathematical services. The general principles are not limited to mathematical services but apply to any web service, however in this study we deal with certain technologies that are specific to the area of online mathematics, in particular OpenMath.

The document is structured following the three main nodes of the architecture: the web server is described in Section 3, the computational engine is described in Section 4, and the client is described in Section 5.

2 Architecture

Figure 1 depicts the various components of the architecture used to provide web services online. The central node is a web server which is enhanced by the capability of handling the simple object access protocol SOAP used by the services it manages. Another node houses the computational engine(s), in this specific example it is the computer algebra system Mathematica. A third node represents the client that accesses the online services either by using a web browser of directly by a command line application. Notice the two types of arrows: the ones with a full arrowhead represent the flow of information between the client and the final computational engine, the other represent input and output flow between the developer of a web service or of a client. In the next sections we will explain the details concerning each node.

3 Web Server

As Figure 1 shows, the web server is the core of the architecture. Such a server provides a container which administers the services that one wants to make available. Administration in this case is very simple, it amounts to being able to list the services, provide a description of their interface and the ability for the administrator of dynamically adding/removing a service. To the final client of a service, the web server provides a URL, which points to somewhere within its domain, where the service access point is located.

*FWF Project P15183 "Brokering of Distributed Mathematical Services"

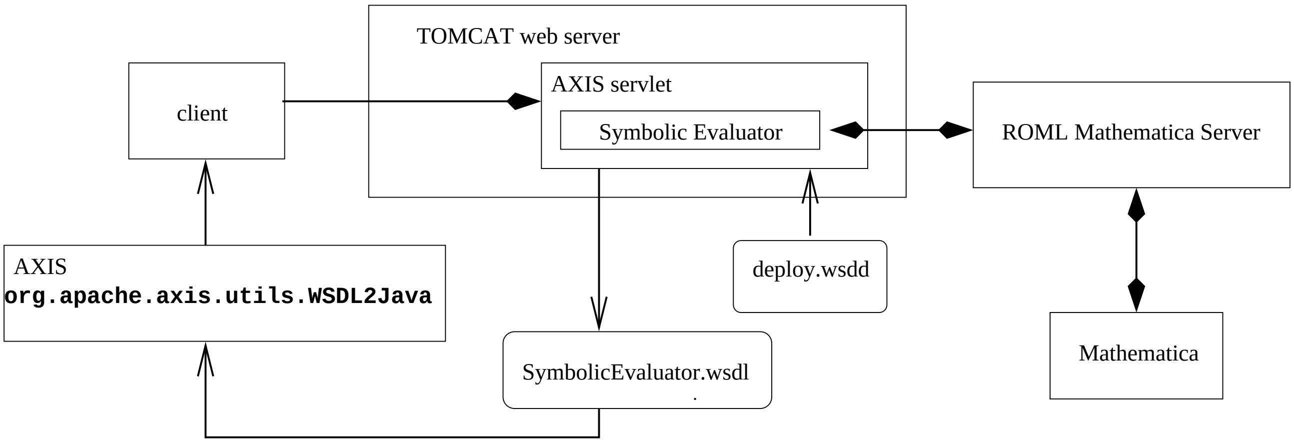

Figure 1: General Architecture

3.1 Installation

At RISC-Linz, we have installed the Apache Tomcat web server on the machine `http://perseus.risc.uni-linz.ac.at:8080`. Tomcat is a highly configurable server that supports JAVA servlets. Currently we have version 4.04 available. The actual code is installed at `$MATHBROKER_SW/jakarta-tomcat-4.0.4-b1-LE-jdk14`.

The web container application that handles web services and SOAP is also an Apache package called AXIS (Apache eXtensible Interaction System). Axis has been installed as a web application of Tomcat just following the instructions distributed with the source code.

The evaluator service is implemented by a JAVA class `Evaluator` that calls the Riaca OpenMath phrasebooks for Mathematica and Gap. The sources are in `$MATHBROKER_HOME/axis_examples/evaluator/service` where it can be compiled using `ant build`. Ant is available among the software installed in `$MATHBROKER_SW` and `$MATHBROKER_SW/jakarta-ant-1.4.1/bin` should be added to the path. In order to deploy the service on the server two steps are needed. The command `ant deploy` will copy the object files to the proper location where axis stores the services (`$TOMCAT_HOME/webapps/axis/WEB-INF/classes`). Then the AXIS `AdminServlet` needs to be informed of the newly available service. This is achieved using a `wsdd` (web service deployment descriptor) file which specifies certain properties of the service.

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
      xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

      <service name="SymbolicEvaluator" provider="java:RPC">
          <parameter name="className" value="Evaluator"/>
          <parameter name="methodName" value="*"/>
      </service>
</deployment>
```

Figure 2: `deploy.wsdd`

After having given this file as input to the AXIS `AdminServlet`[1], the service is online at a certain url, in this case `http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator`.

4 Computational Engines

The computational engine in this architecture is the computer algebra system Mathematica accessed through the JAVA wrapper provided by the Riaca OpenMath library. The wrapper takes input in Mathematica syntax and

[1] It is easily done using the command `$MATHBROKER/bin/axis deploy.wsdd`

returns output in Mathematica syntax again. Currently, for testing purposes, we are using a Mathematica server running remotely on a machine at the Technical University of Eindhoven. In the near future we plan to have a Mathematica server running locally at RISC.

5 Clients

A client is a piece of software that connects to the service. There are two possible ways in which to access the service: from the command line or using a browser as a graphical user interface. The provider of a service might wish to also provide an appealing user interface to access the service. In particular for services that expect data in XML, it is a better solution to have the client produce the XML from a more natural, user-friendly input. This is especially true for mathematical input which can be asked in a structured way and then internally organized as required by the service.

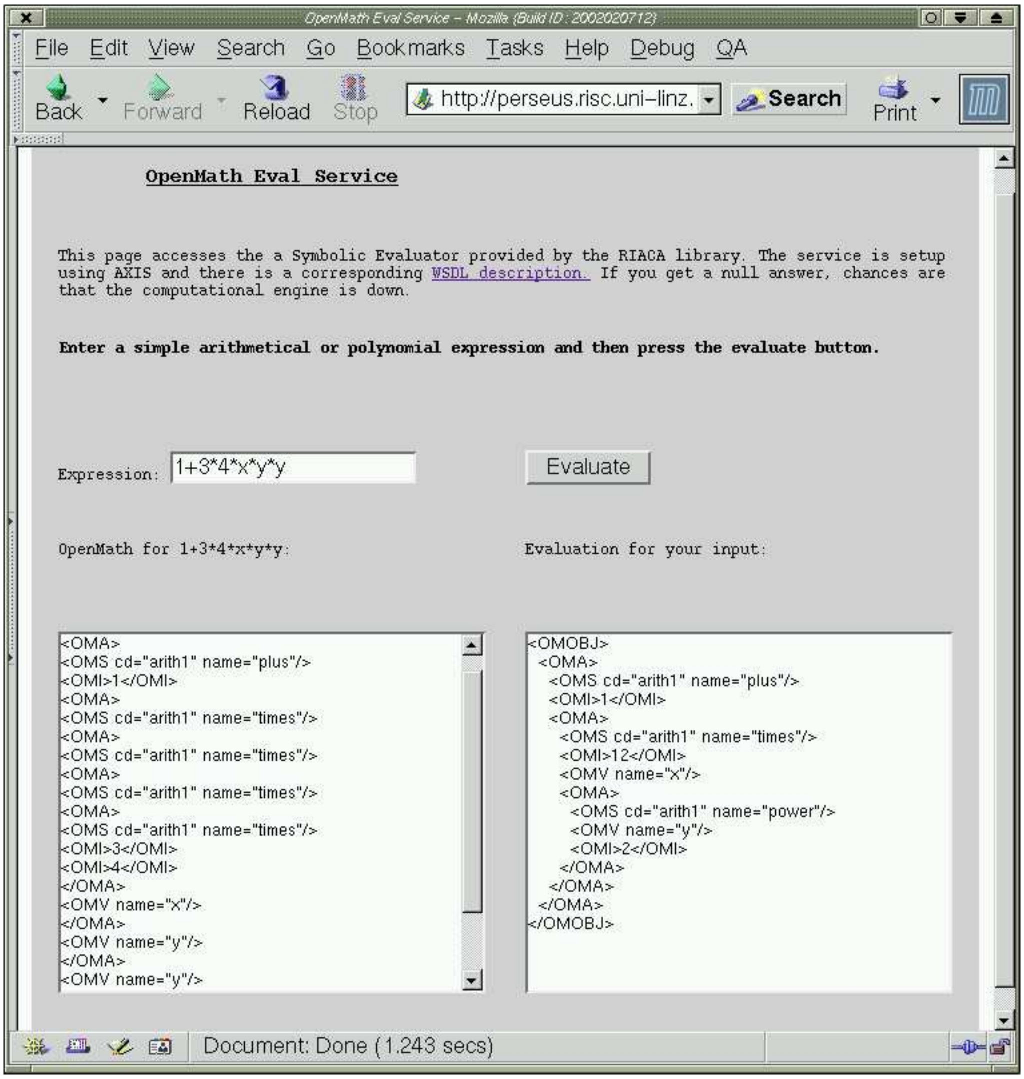

Figure 3: Screenshot of the Symbolic Evaluator client

5.1 Installation

The client we have implemented as a web application takes as input an expression, parses it and generates an OpenMath object in XML encoding for it. This object is then the argument sent to the `SymbolicEvaluator` service which takes care of decoding it to the proper syntax, depending on the selected computational engine. There are three main components of the client:

Parsing The parser is generated by using a parser generator called `antlr`. Sources for the parser are in `$MATHBROKER_HOME/axis_examples/evaluator/parser`.

GUI A jsp (JAVA server page) page acts as GUI for accessing the service. The code is contained in a single file `$MATHBROKER_HOME/axis_examples/evaluator/webapp/eval.jsp` and is deployed by the command `ant deploy` to `$TOMCAT_HOME/webapps/openmath`.

Service Connection The code for the classes that allow to connect to the service have been generated in `$MATHBROKER_HOME/axis_examples/evaluator/webapp/src` and can be compiled or deployed by the commands `ant build` and `ant deploy` respectively. Deployment amounts to copying the class object files to `$TOMCAT_HOME/webapps/openmath/WEB-INF/classes`.

AXIS provides the application `org.apache.axis.utils.WSDL2Java` to generate the classes needed by a client from the WSDL description of the service.The classes can be easily generated using the command

`$MATHBROKER/axis_examples/bin/wsdl2j http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator?wsdl`.

Such a description is obtained easily by adding `?wsdl` to the url of the service, namely `http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator?wsdl`.

Although in our implementation we have installed the client on the same machine/server where the service resides, it could also have been otherwise installed on a different machine/server.

A different client coded following the examples in the AXIS user-guide, that can be used from the command line, is available in the directory `$MATHBROKER_HOME/axis_examples/evaluator/client`. The command to run uses a script in order to setup the CLASSPATH correctly and is invoked as shown below, where the second argument is a file which contains the OpenMath XML encoding of the expression being evaluated.

```
 [perseus:client!114] test Mathematica 1+11
<OMOBJ>
  <OMA>
    <OMS cd="arith1" name="plus"/>
    <OMI>1</OMI>
    <OMI>11</OMI>
  </OMA>
</OMOBJ>

Returned result:<OMOBJ>
  <OMI>12</OMI>
</OMOBJ>
```

6 Conclusion

We have documented the implementation and installation of a very general mathematical web service consisting in a symbolic evaluator. Although quite simple, yet this example can be considered the basis on which to construct more involved/specific services. We have also mentioned several technologies that were heavily used to carry out parts of the development: web server technologies, such as jsp and SOAP, mathematical communication technologies, such as OpenMath and OpenMath Phrasebooks, and JAVA technologies like `antlr`. The combination of all of these made the overall job quite simple.

We are still missing, mainly due to lack of time, a good way of rendering the resulting OpenMath object possible by using MathML-presentation, and a connection to more computational engines. We can already connect to the computational group theory package GAP without any modification to our code beside setting up the exact location of a GAP engine.

A Environment variables

```
JAVA_HOME=/home/scratch/j2sdk1.4.0
MATHBROKER_HOME=/home/mathbroker/
MATHBROKER_SW=/home/mathbroker/mathbroker_sw
JWSDP_HOME=/home/mathbroker/mathbroker_sw/jwsdp-1_0-ea1
TOMCAT_HOME=/home/mathbroker/mathbroker_sw/jakarta-tomcat-4.0.4-b1-LE-jdk14
AXIS_HOME=/home/mathbroker/mathbroker_sw/xml-axis-beta1-rc2
```

B Symbolic Evaluator WSDL

Here below is included the WSDL specification which is obtained by querying the online service `http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator?wsdl`.


```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
            targetNamespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
            xmlns="http://schemas.xmlsoap.org/wsdl/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:impl="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator-impl"
            xmlns:intf="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:message name="WithGapResponse">
    <wsdl:part name="return" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="WithMathematicaRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="WithMathematicaResponse">
    <wsdl:part name="return" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="WithGapRequest">
    <wsdl:part name="in0" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="Evaluator">
    <wsdl:operation name="WithMathematica" parameterOrder="in0">
      <wsdl:input message="intf:WithMathematicaRequest"/>
      <wsdl:output message="intf:WithMathematicaResponse"/>
    </wsdl:operation>
    <wsdl:operation name="WithGap" parameterOrder="in0">
      <wsdl:input message="intf:WithGapRequest"/>
      <wsdl:output message="intf:WithGapResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SymbolicEvaluatorSoapBinding" type="intf:Evaluator">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="WithMathematica">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input>
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
                       use="encoded"/>
      </wsdl:input>
      <wsdl:output>
        <wsdlsoap:body  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
                        use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="WithGap">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input>
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
                       use="encoded"/>
```

```
      </wsdl:input>
      <wsdl:output>
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"
                       use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="EvaluatorService">
    <wsdl:port binding="intf:SymbolicEvaluatorSoapBinding" name="SymbolicEvaluator">
      <wsdlsoap:address location="http://perseus.risc.uni-linz.ac.at:8080/axis/services/SymbolicEvaluator"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```