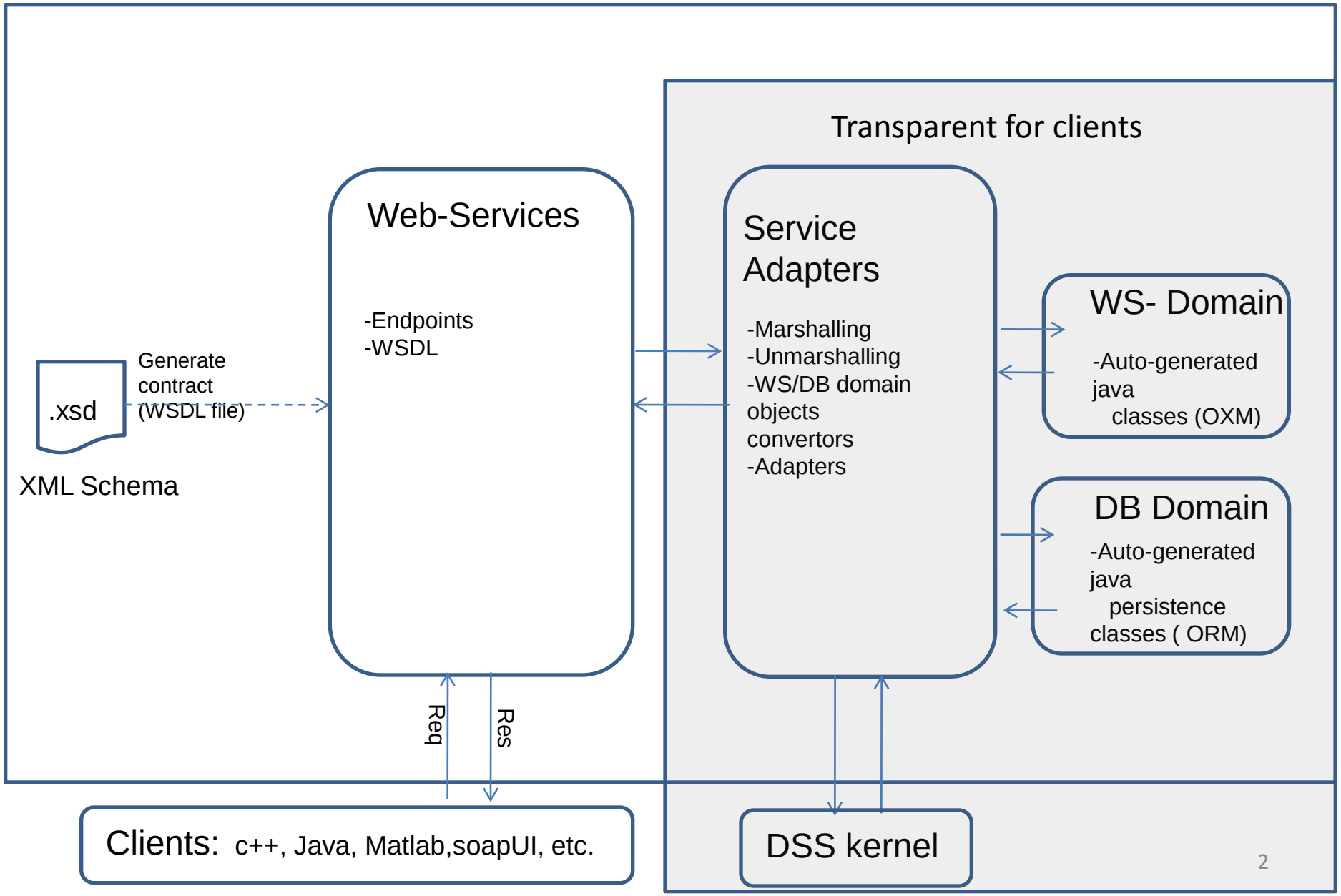
SMS Client Examples

Hongtao Ren, Marek Makowski

Outline:

- Overview of WS
- SMS XML schema and WSDL
- Exploring the WS
- C++ client example
- Java client example

Overview

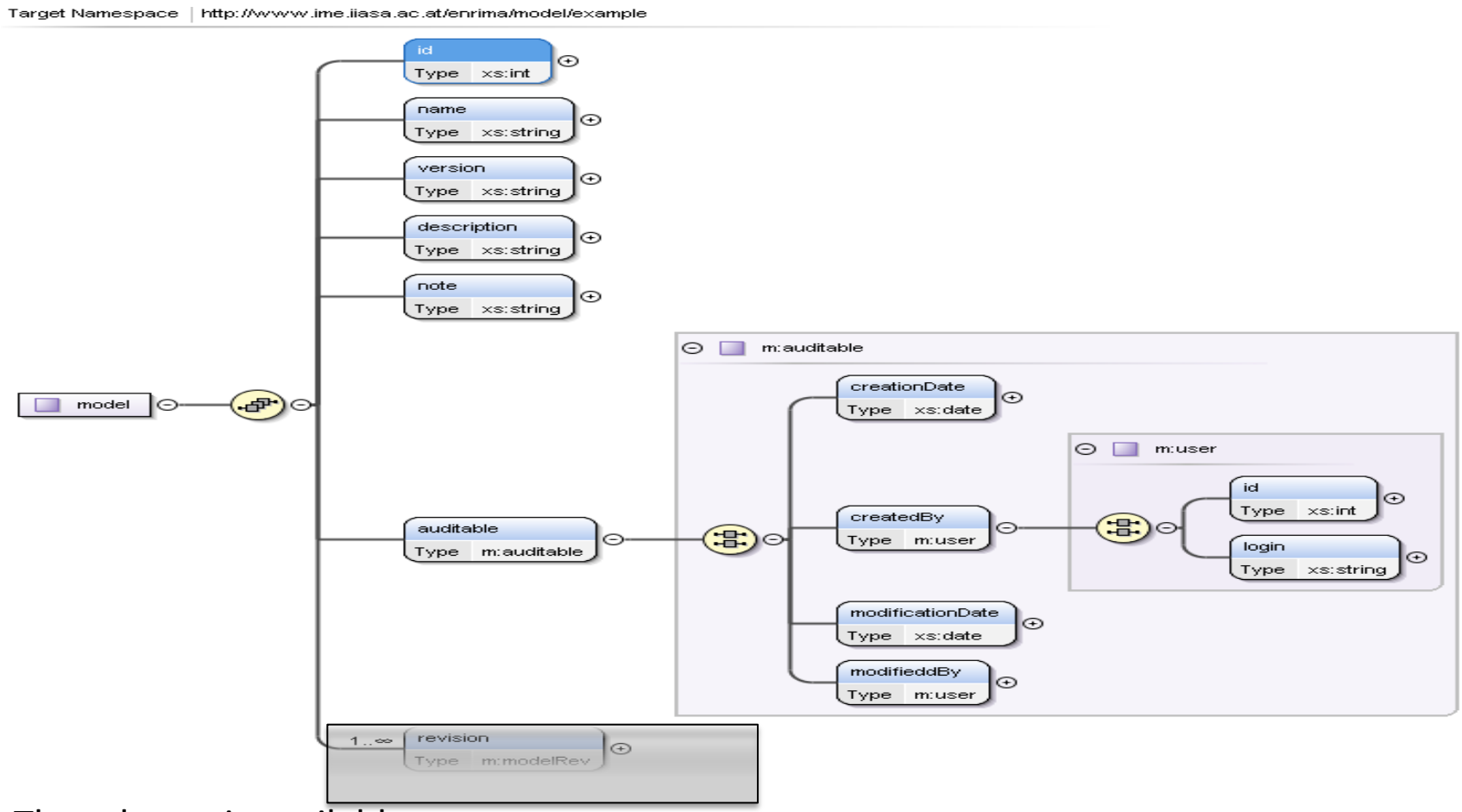

Transparent for clients
Web-Services
-Endpoints
-WSDL
Service Adapters
-Marshalling
-Unmarshalling
-WS/DB domain objects convertors
-Adapters
WS- Domain
-Auto-generated java classes (OXM)
DB Domain
-Auto-generated java persistence classes (ORM)
.xsd
Generate contract (WSDL file)
XML Schema
Req
Res
Clients: c++, Java, Matlab,soapUI, etc.
DSS kernel

XML schema of SMS

Roles of XML schema:

1) Defines structure and content of data exchanged between modules (DSS engine, solver, UI)
2) The basis for WSDL
3) The basis for interface classes for client application

XML schema (SMS example)



The schema is available at:
https://github.com/h-t-ren/enrima-ws-server/blob/master/docs/model.xsd
Documentation of the schema:
https://raw.github.com/h-t-ren/enrima-ws-server/master/docs/model.pdf

A snippet of WSDL

```
< wsdl:portType name="enrima">
        <wsdl:operation name="getModelDescription">
          <wsdl:input message="tns:getModelDescriptionRequest“name="getModelDescriptionRequest">
         </wsdl:input>
        <wsdl:output message="tns:getModelDescriptionResponse" name="getModelDescriptionResponse">
        </wsdl:output>
     </wsdl:operation>
      <wsdl:operation name="saveModelDescription">
       <wsdl:input message="tns:saveModelDescriptionRequest" name="saveModelDescriptionRequest">
      </wsdl:input>
      <wsdl:output message="tns:saveModelDescriptionResponse" name="saveModelDescriptionResponse">
     </wsdl:output>
  </wsdl:operation>
   <wsdl:operation name="getModelList">
    <wsdl:input message="tns:getModelListRequest" name="getModelListRequest">
   </wsdl:input>
   <wsdl:output message="tns:getModelListResponse" name="getModelListResponse">
   </wsdl:output>
 </wsdl:operation>
</wsdl:portType>
```

The WSDL is available at:
http://www.ime.iiasa.ac.at/enrima_ws_tst1/enrima.wsdl

Exploring WS

Testing WS is optional but recommended to check:

1) If the message structure and content fit application domains?
2) If the operations (functions) fit functional requirement?
3) If the performance fit non-functional requirements?

Tools

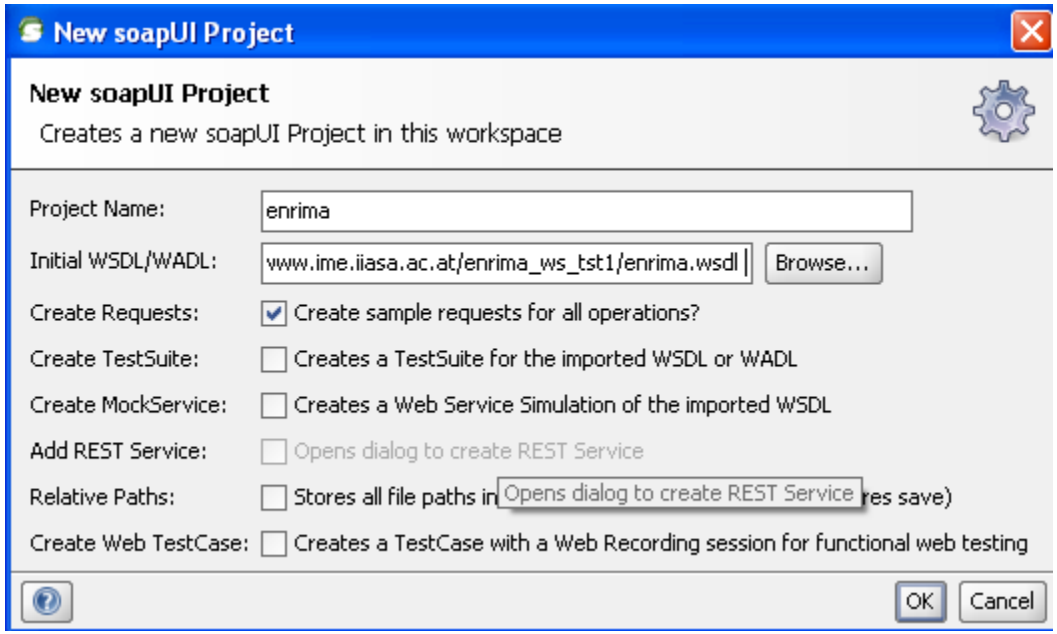
1) soapUI
2) Web King
3) XML spy

…

Exploring WS: soapUI (1-2)

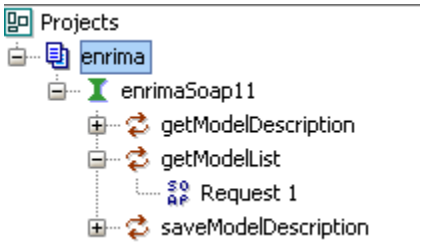
1） Download and install soapUI (http://www.soapui.org/)

2） File -> New soapUI project, in the intial WSDL/WADL field, Enter http://www.ime.iiasa.ac.at/enrima_ws_tst1/enrima.wsdl and click OK

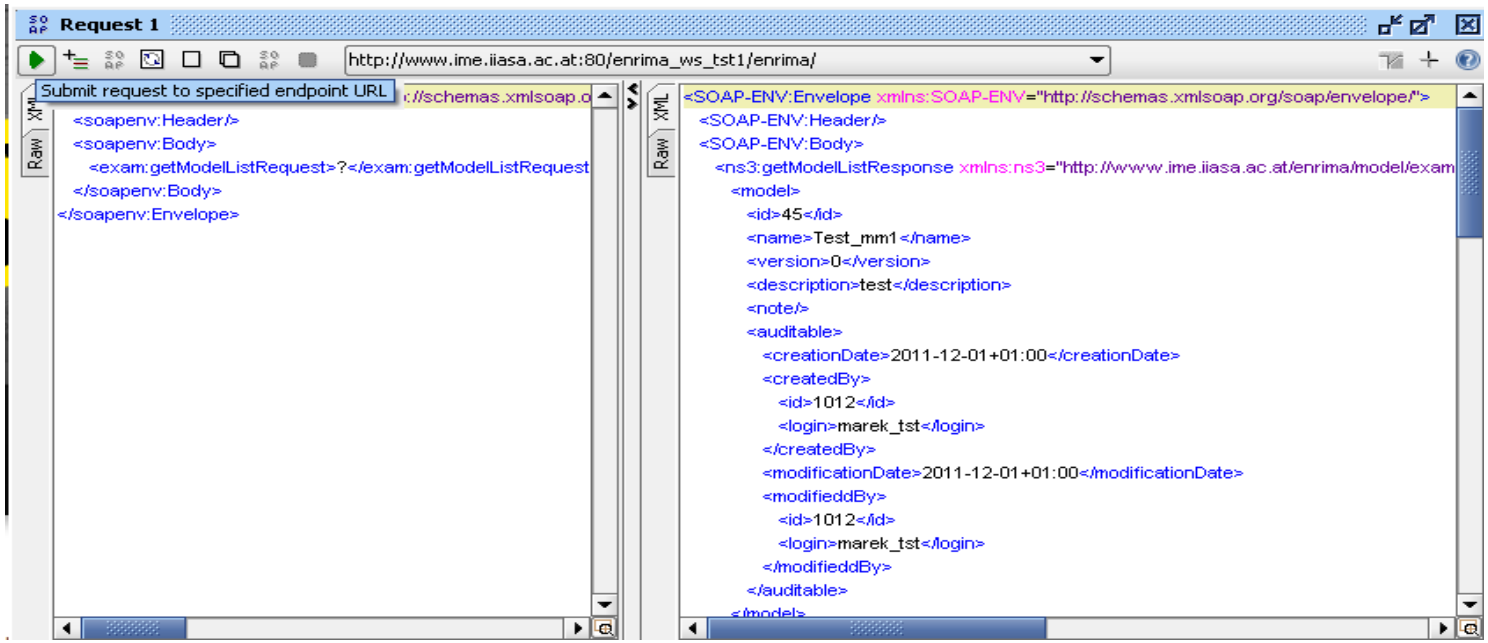


Exploring WS: soapUI (3-4)

3) You should now see that the WSDL was successfully added to the project by seeing the operations in the Web Service in the navigator



4) Click on "getModelList"; double click on "Request 1" and then click the green triangle to submit the request. The result of the response will be shown in the right pane.



A simple C++ client

Download :

- git clone git://github.com/h-t-ren/enrima-ws-gsoap-client.git
- or https://github.com/h-t-ren/enrima-ws-gsoap-client , click ‘zip’ icon

Files:

- main.cpp: a simple client, will be replaced by real applications
- stdsoap2.h, stdsoap2.cpp: from gsoap
- soapH.h, soapC.cpp, soapStub.h, soapenrimaSoap11Proxy.h, soapenrimaSoap11Proxy.cpp: generated by gsoap

Compile and run the example:

```
g++ -o enrima *.cpp
or:  g++ -o enrima *.cpp -lsocket -lnsl
./enrima
```

gSoap (Optional)

About gSoap:
gSoap: The gSOAP toolkit is an open source C and C++ software development toolkit for SOAP/XML Web services and generic (non-SOAP) C/C++ XML data bindings. The toolkit analyzes WSDLs and XML schemas (separately or as a combined set) and maps the XML schema types and the SOAP messaging protocols to easy-to-use and efficient C and C++ code.

You can create files by the following:

1) Download gsoap from http://sourceforge.net/projects/gsoap2/files/gSOAP/, please select gsoap_2.8.3.zip or above
2) Unzip and install gsoap (assume you unzip it to $HOME/gsoap-2.8)

```
cd $HOME/gsoap-2.8
./configure —prefix=$HOME/gsoap
make & make install
```

3) Create enrima client test

```
mkdir tst
cp $HOME/gsoap-2.8 /gsoap/{typemap.dat,stdsoap2.{h,cpp}} tst
```

4) Generate enrima.h

```
$HOME/bin/wsdl2h -o enrima.h -t typemap.dat http://www.ime.iiasa.ac.at/enrima_ws_tst1/enrima.wsdl
```

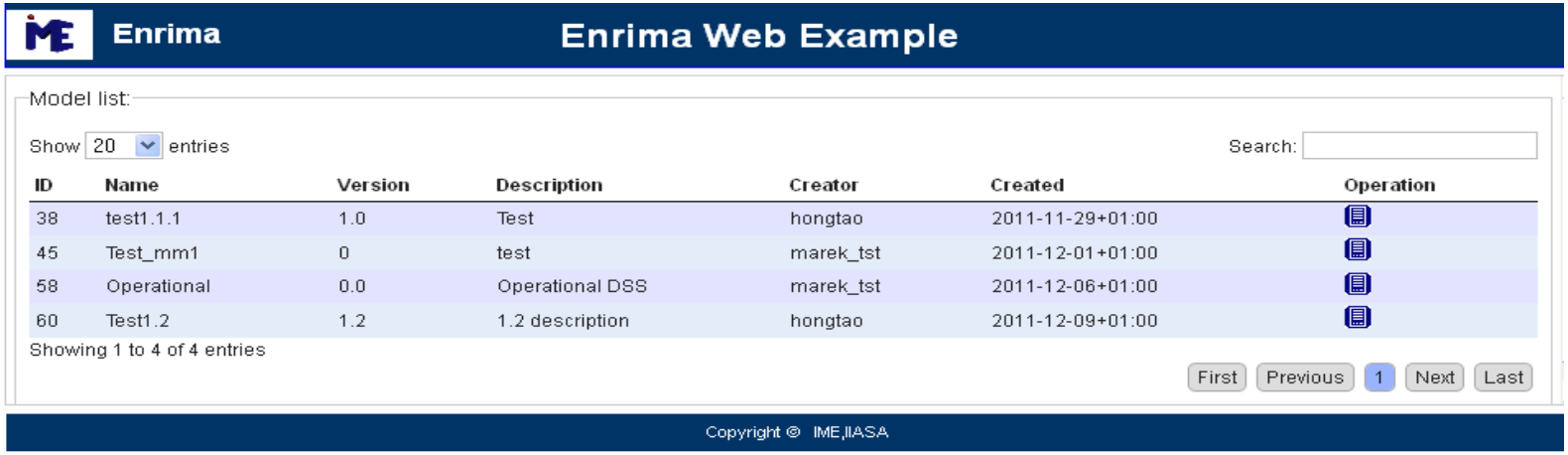
5) Generate the Client-site *.h & *.cpp files

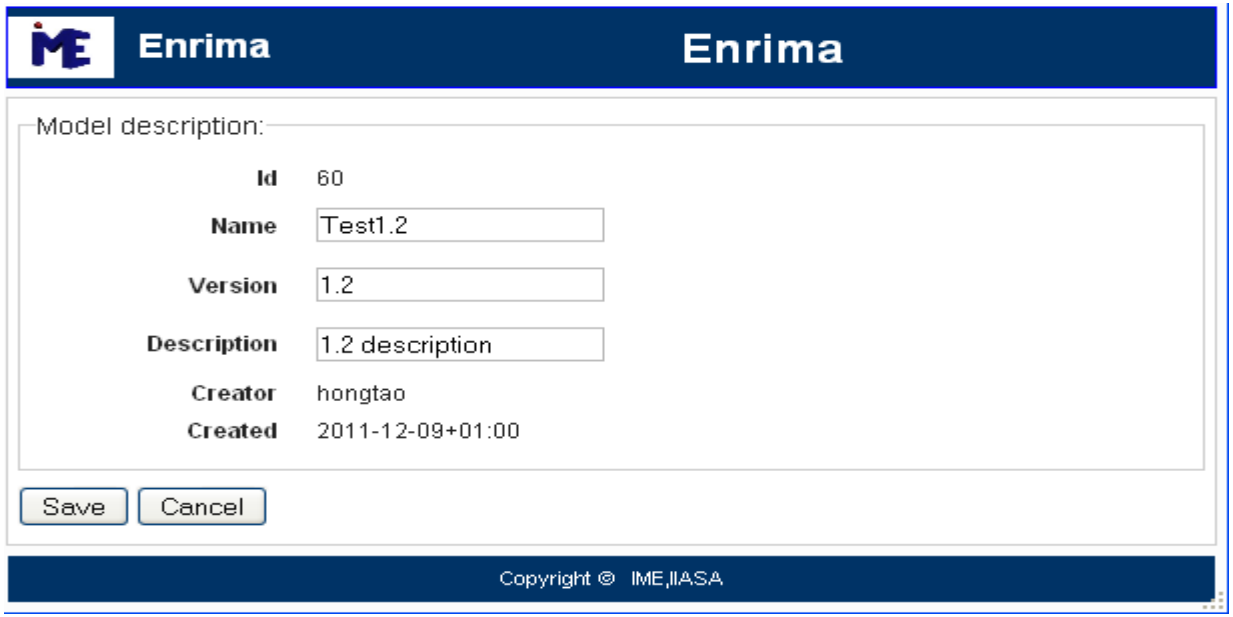
```
$HOME/bin/soapcpp2 -C -i enrima.h -I $HOME/gsoap/share/gsoap/import/
```

Java client

1)Install development environments (JDK 6 or above, Maven 3 or above)
2)Install Git (optional)
3) Download the enrima-ws-java-client example:
 git clone git://github.com/h-t-ren/enrima-ws-java-client.git
 or https://github.com/h-t-ren/enrima-ws-java-client and click "zip" button
4) Install:
 $cd enrima-ws-java-client
 $mvn clean install
5) Run tomcat
 $cd ui
 $mvn tomcat:run
 (to stop tomcat: ctrl+c)
6) webpage: http://localhost:8080/enrima-web/

Java client web





Resources and Tools

-XML tutorial: http://www.w3schools.com/xml/default.asp

-XML schema tutorial: http://www.w3schools.com/schema/default.asp

-WSDL tutorial: http://www.w3schools.com/wsdl/default.asp

-soapui: http://www.soapui.org/

-gSoap: http://www.cs.fsu.edu/~engelen/soap.html

-Maven: http://maven.apache.org/

-Git: http://git-scm.com/

-Matlab: http://sipi.usc.edu/manuals/matlab701/techdoc/matlab_external/ch_soap4.html

Summary

Main features:

- Modules on heterogeneous platforms (OS, languages)
- WS easy to use
  - XML knowledge not need,
  - Generated C++/Java objects can be embedded into application

In-depth discussion necessary:

- Each module’s requirements for data structure and WS functionality
- Structure of SMS (implies XML schema)
- Effectiveness of WS (and the DW)