PUPPET - User Manual

Guglielmo De Angelis

March 4, 2008

Abstract

This article is the user's guide for PUPPET . Please refer to [3] for the detailed description of the whole approach, the architectural description of the proposed implementation, the tools and the standard that have been used, or for any kind of motivation of the work.

Contents

1 Puppet Overview

To ensure consistent cooperation for business-critical services, with contractually agreed levels of Quality of Service, SLA specifications as well as techniques for their evaluation are nowadays irremissible assets. Puppet (Pick UP Performance Evaluation Test-bed) is an original approach developed within PLASTIC for the automatic generation of test-beds to empirically evaluate the QoS features of a Web Service under development. Specifically, the generation exploits the information about the coordinating scenario, the service description (WSDL) and the specification of the agreements (WS-Agreement).

As described in [3, 2, 1], PUPPET was originally designed in order to automatically generate stubs conforming only to SLA behaviors ignoring functional aspects. In other words, if invoked the stub were able to provided good QoS values but the responses were not built to be semantically meaningful (i.e. always returning constant values). However, in the general case extra-functional aspects are tightly coupled with functional characteristics. The current version of PUPPET integrates the emulation of the functional specifications as part of the generated testbed. The obtained environment can expose not only the specified extra-functional parameters but also meaningful functional behavior. Specifically, PUPPET generates stubs for Web Services which respect both an extra-functional contract expressed via a Service Level Agreement (SLA), and a functional contract modeled via a Service State Machine (SSM, see Chapter 4.6).

2 Technical info

Provider CNR

Introduction PUPPET is a tool for the automatic generation of test-beds to empirically evaluate the QoS features of a Web Service under development. The stubs generated with PUPPET conform both the extra-functional contract expressed via a Service Level Agreement (SLA), and to the functional contract modeled via a state machine.

Development status Version PuppetD4.3

Intended audience Developers who intend to test a PLASTIC service in composition with 3rd party Web Services

License Open source under GPLv3 with some exceptions to include libs

Language Java, XML

Environment (set-up) In the following the required software and hardware :

    Hardware: No specific hardware is required.

    Software: The following JAR library are required in order to compile and to launch PUPPET :

        Apache Axis 1.4 Libs : axis-ant.jar, axis.jar, commons-discovery.jar, commons-logging.jar, jaxrpc.jar, jsr173_1.0_api.jar, log4j-1.2.8.jar, wsdl4j.jar, saaj.jar – availabe at http://ws.apache.org/axis.

        Apache XMLBeans Libs : xbean.jar – availabe at http://xmlbeans.apache.org.

        String Template Libs : stringtemplate.jar – availabe at http://www.stringtemplate.org.

        INI4J Libs : ini4j.jar, ini4j-compat.jar – availabe at http://ini4j.sourceforge.net.

        Jambition Libs : SSMSimulator.jar, minerva.jar (see Chapter 4.6)

        Other Libs : antlr-2.7.7.jar, xercesImpl.jar, xmlsec.jar, mail.jar, activation.jar, java-getopt-1.0.13.jar

        Please note that **PuppetD4.3.tgz** archive includes a version of these libraries in the directory **PuppetD4.3/externalLibs**.

Platform Java jdk1.6 or later

Download Download the official version of PUPPET in PLASTIC at http://plastic.isti.cnr.it/download/tools

Documents Related documents on the approach, the architectural description, and the implementation of PUPPET are [3, 2, 1].

Tasks N/A

Bugs N/A

Patches N/A

Contact guglielmo.deangelis@isti.cnr.it, andrea.polini@isti.cnr.it

3 Deployment

3.1 Install

Unzip the archive **PuppetD4.3.tgz** and configure the environmental CLASSPATH with the required libs indicated above.

The directory structure is the following:

- PuppetD4.3
    - doc
    - example
    - externalLibs
    - puppet.jar
    - puppetLibs
    - puppet.sh
    - runPuppet.bat
    - src
    - xml

The directory **PuppetD4.3**/**doc** contains this manual. The directory **PuppetD4.3**/**xml** contains the definitions mapping of the WS-Agreement statements into the Java code. The directories **PuppetD4.3**/**externalLibs** and **PuppetD4.3**/**puppetLibs** contains the libraries required by PUPPET in order to run. As your preference, you would append the name of the **.jar** files in **PuppetD4.3**/**puppetLibs** and **PuppetD4.3**/**externalLibs** to the Java CLASSPATH variable. **PuppetD4.3**/**puppet.sh** and **PuppetD4.3**/**runPuppet.bat** are exacutable batch scripts that set the Java CLASSPATH variable and run PUPPET on a given input file.

3.2 Configure

PUPPET generates stubs for web services according to what defined in a given configuration file. The configuration file is supposed to compile with the standard INI File Format. In such file, PUPPET looks for the section named **[mainSection]**. PUPPET loads its parameters as specified in the configuration held by this section.

The parameters that could be specified into the input configuration file are:

wsdlPath : It is the path to the directory holding the WSDL specifications of all the services whose emulators would be generated by means of PUPPET . Required.

targetPath : It is the path to the directory where PUPPET will dump the generated stubs. Required.

JarPath : It is the path to the required JAR file libraries listed above.[1]. Required.

wsaFilename : It is the name of the WS-Agreement file describing the agreements among the considered services. If it is not specified, PUPPET would look for a file named: **agreement.xml**. Optional

---

[1]In the future releases it would be deprecated

trueTermsFilename : It is the name of the file holding the terms of the agreement that have to be considered as fulfilled. As described in Sec 4.1, for each term in the agreement, PUPPET will generate code that emulates an extra functional behavior if and only if the term is fulfilled. If this file name is not specified, PUPPET would look for a file named: **gtTrueItemList.xml**. Optional

wsaPath : It is the path to the directory holding the WS-Agreement file. Required.

trueTermsPath : It is the path to the directory holding the *trueTermsFilename*. If it is not specified, PUPPET would assign to this parameter the path to the directory holding the WS-Agreement file (*wsaPath*). Optional.

qcMappingFilename : It is the path to the file holding the template mapping of the Qualifying Conditions in WS-Agreement on to the the Java code that will be generated. PUPPET already includes a predefined mapping file. Even though it is possible to change this mapping, we strongly discourage from changing it. Optional.

sloMappingFilename : It is the path to the file holding the template mapping of the Service Level Objectives in WS-Agreement on to the Java code that will be generated. PUPPET already includes a predefined mapping file. Even though it is possible to change this mapping, we strongly discourage from changing it. Optional.

ambitionMode : If it is set to "**on**" enables the emulation of the functional behavior with Jambition. By default it is set to "**off**". Optional.

3.3 Usage

Let us assume that the variable **CLASSPATH** of the JVM you are executing includes both the JAR files listed in the item **Tools** above, and those contained in **PuppetD4.3/puppetLibs**. PUPPET usage is:

java -cp $CLASSPATH:puppet.jar puppet.Puppet <IniConfigurationFile>

An alternative way to run PUPPET is executing the batch scripts **PuppetD4.3/puppet.sh** and **PuppetD4.3/runPuppet.bat** [2] on a given <IniConfigurationFile>.

4 Tutorial

4.1 Terms in the Agreement and Generation Process

In WS-Agreement [8], an agreement specification is composed by one or more terms. These terms are grouped in a logic formula by means logic connectors (**All** – logic AND, **OneOrMore** – logic OR, and **ExactlyOne** – logic XOR).

Different scenarios could be considered defining a set of terms in the agreement that are assumed fulfilled. This set of terms enables in PUPPET the generation of the Java code emulating the extra functional behavior. PUPPET loads the list of these terms parsing an XML file. The value of each element in the XML file refers to the name of a term in the WS-Agreement specification. The example in Table 1 shows how to enable the code generation for the term named **WarehouseGT** in the agreement specification.

4.2 The Syntax for the Terms in the WS-Agreement Contracts

This section describes the domain specific syntax adopted in order to instantiate the generic contents defined by the Terms in WS-Agreement. The section is organized in tree main parts: Section 4.2.1 introduces the syntax that PUPPET uses in order to specify under which conditions a Term is applicable. Section 4.2.2 introduces the syntax that PUPPET uses in order to specify the extra-functional property the Term predicates about. Section 4.2.3 introduces the syntax that PUPPET uses in order to limit the scope of a Term only to some operations among all the ones that a Service exports.

---

[2]Respectivelly under Unix-like and Windows operating systems

```
1  <wsag:AgreementOffer xsi:schemaLocation="..." xmlns:wsag="...">
2  ...
3      <wsag:GuaranteeTerm wsag:Name="WarehouseGT"
4        wsag:Obligated="ServiceProvider">
5        ...
6      </wsag:GuaranteeTerm>
7  ...
8  </wsag:AgreementOffer>
```

```
1  <tns:TrueGTList xmlns:tns="..." xmlns:xsi="..." xsi:schemaLocation="...">
2  ...
3    <tns:GTItemName>WarehouseGT</tns:GTItemName>
4  ...
5  </tns:TrueGTList>
```

Table 1: Enabling the code generation in PUPPET
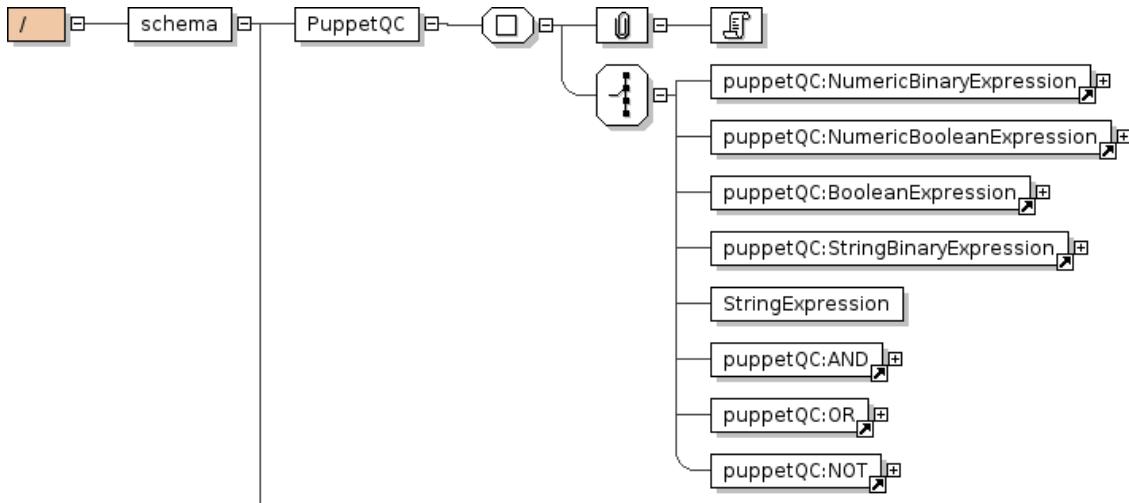
Figure 1: Expressions in the Qualifying Condition

4.2.1 Qualifying Conditions

In WS-Agreement the Qualifying Conditions of a Term may appear to express a precondition under which a Term holds [8]. In PUPPET , a Qualifying Condition can be formulated in terms of atomic expressions typed as Numeric, Boolean, or String (see Figure 1). The atomic expressions can be combined by means of the Boolean operators: AND (see 2), OR (see 3), and NOT (see 4).

Figure 5 depicts the elements that can be used in order to construct an atomic expressions. Also, for each operation type, it shows the operators supported in this release.

4.2.2 Service Level Objective

The specification of WS-Agreement defines the Service Level Objective (of type xsd:anyType), as the element expressing the condition that must be met to satisfy the guarantee Term [8].

This version of PUPPET handles constraints on the maximum admissible response time (i.e. service latency) and constraints on reliability (see Figure 6).

The time elapsed by a service when invoked (latency) is defined specifying the maximum admissible response time and a probability function describing how the delays are distributed. In this version, is it possible to define delays that are normally distributed or that follow the Poisson's law.

The constraints on the reliability of a Service are defined in terms of the maximum number of failure (**ReliabilityPerc** in Figure 6) in a given window of time. Also in this case, this release offer to describe the distribution of the failures in the window either as normal or as Poisson's.
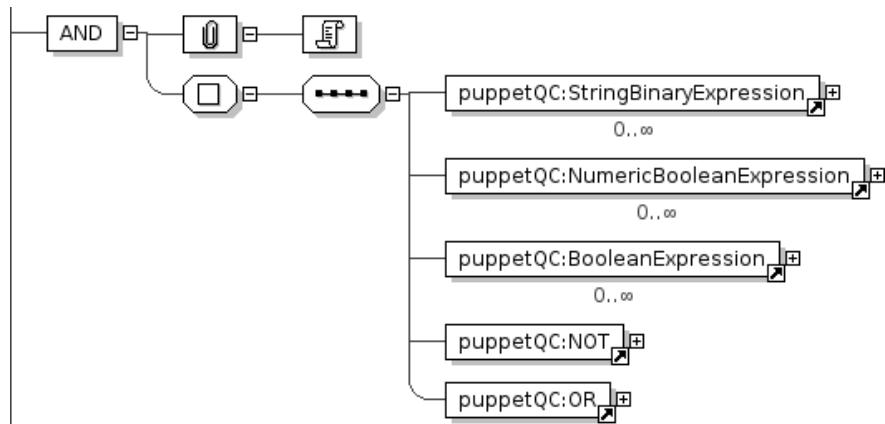
Figure 2: AND in the Qualifying Condition
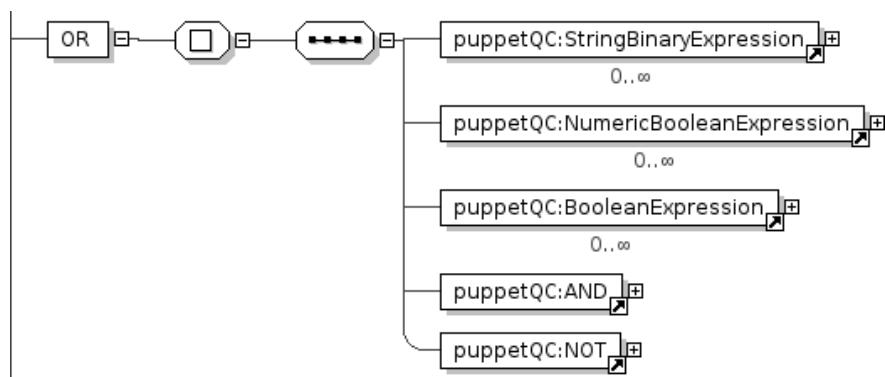
Figure 3: OR in the Qualifying Condition
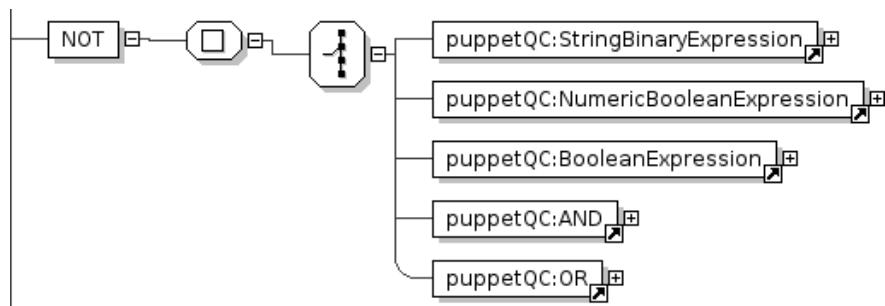
Figure 4: NOT in the Qualifying Condition
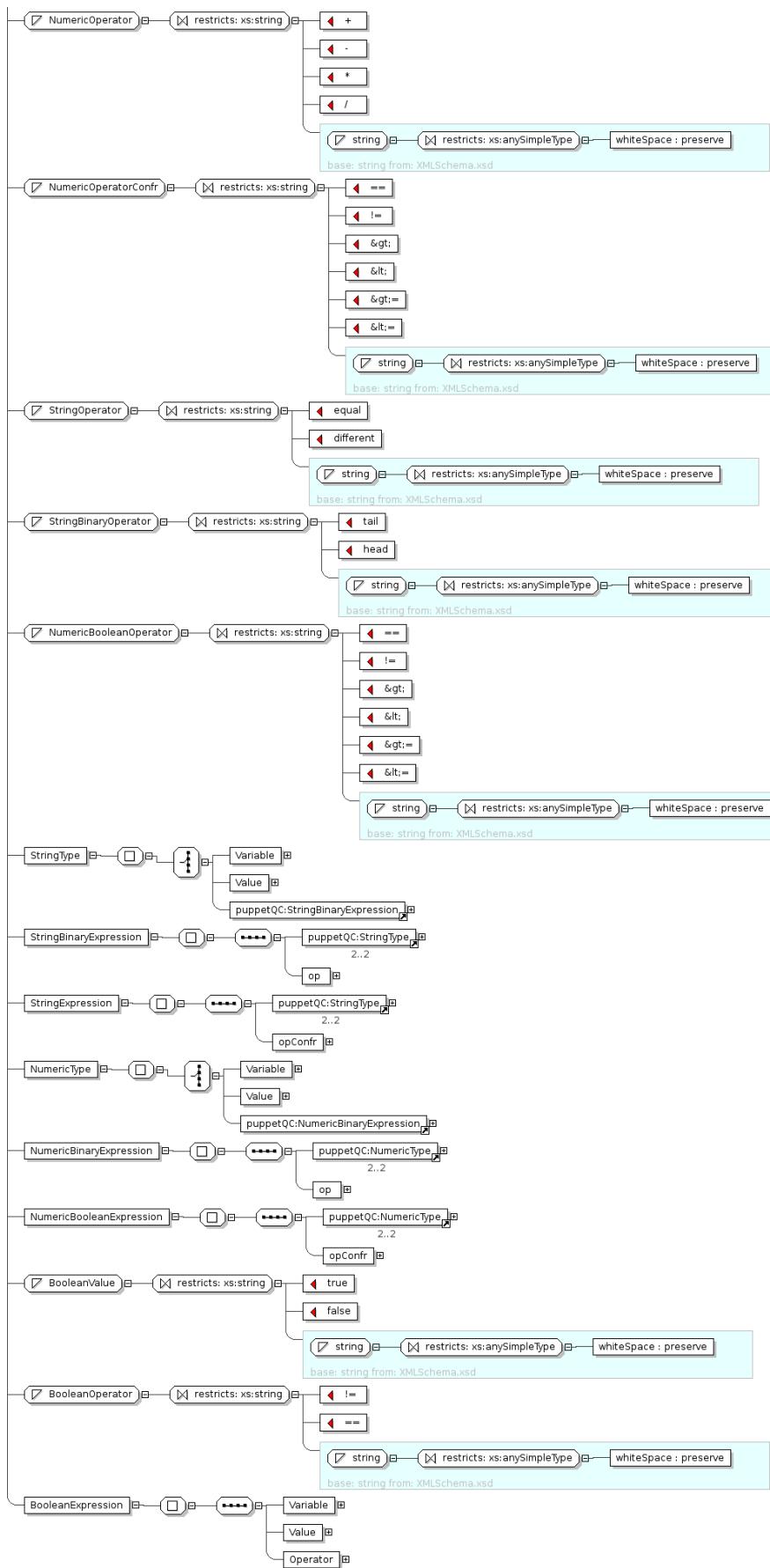
Figure 5: Operators in the Qualifying Condition

7
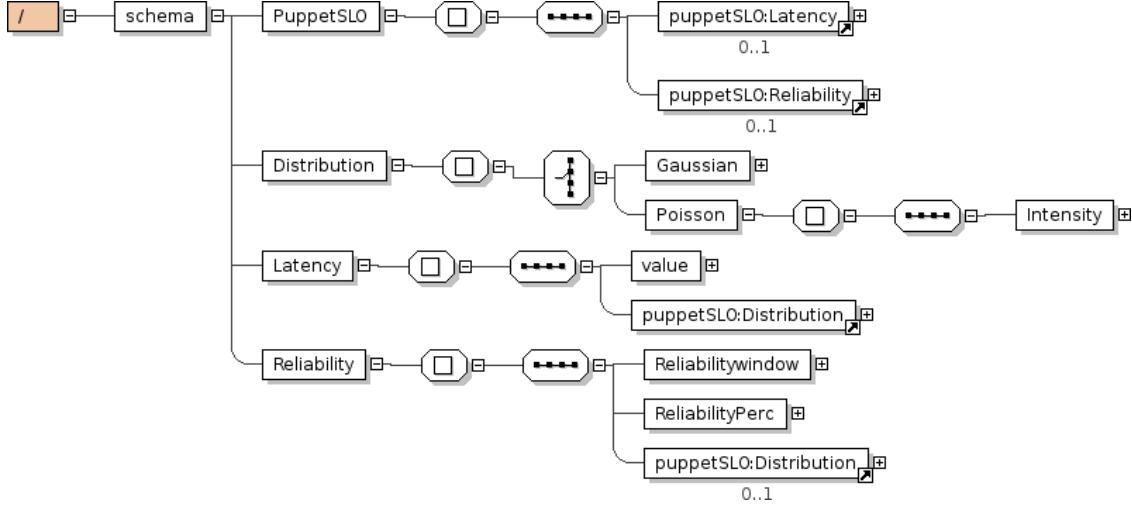
Figure 6: Extra-Functional Properties in the Service Level Objective
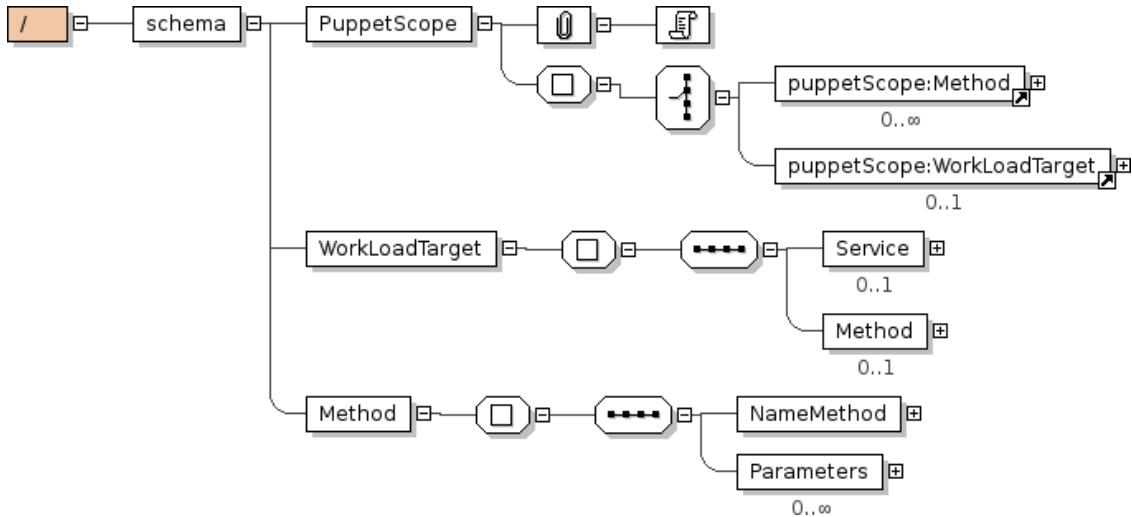
Figure 7: Defining the Scope of a Term

4.2.3 Scope

The scope of a Term describes to what service element specifically a term applies. For example, a term might only apply to one operation of a Web service at a particular end point. According to the specification of WS-Agreement [8], the scope of a Term contains a **ServiceName** attribute and any other XML structure describing a sub-structure of a service to which the scope applies.

In this version of PUPPET it is possible to define the list of the operations affected by a specific Term as depicted in Figure 7. If a Term does not specify any scope, PUPPET would generate the emulation of the extra-functional property in all the operations exported by the service the Term refers to.

4.3 Writing an Agreement

In PLASTIC there are two possible way to write WS-Agreement specification for PUPPET . The former is to write it directly according to the indications given in [3]. The latter is to exploit the PLASTIC's editor of SLA as explained in the following.

The PLASTIC conceptual model [7] defines the reference SLA concepts adopted the in the project. This means that the specific implementations of the various environments should consider to manage at least the QoS annotations expressed in [7] and then refined in D1.2 [6].

According to the conceptual model, [5] defines with SLAng [9] an abstract syntax for the agreements. Such syntax would be instantiated in several concrete syntax. Each concrete syntax refers to a given kind of specification. For example in [5] the SLAng concepts were expressed using the HUTN (Human-Usable Textual Notation) as a concrete syntax.

The concrete syntax of SLAng could also refer to other languages for SLA specification. In that sense, a WS-Agreement specification could be seen as a concrete instantiation of the SLAng's abstract syntax. Note that such association is valid under the assumption that the two specifications predicate on the same kind of concepts.

In deliverable D2.2, the consortium presents a tool support for SLAng. It is an Eclipse-based editor for SLAng, in the form of an Eclipse plugin. The joint work between WP2 and WP4 developed an extension to the SLAng editor including a syntactic translation engine that generates WS-Agreement specification. The output produced by the plugin extension of the SLAng editor could be used as input for PUPPET .

4.4 Functional Behavior with Jambition

The integrated work of the team developing PUPPET and the team developing Jambition (see Chapter 4.6) in WP4 included in PUPPET (version PuppetD4.3) the features to generate stubs whose behavior conforms to both extra-functional contracts and a functional specifications.

As reported in Chapter 4.6, the functional behavior of a service in Jambition is modeled using a state machine called *Service State Machine* (SSM).

Enabling the **ambitionMode** in the INI configuration as specified in Section 3.2, PUPPET would include in the generated stubs the code emulating the functional behavior.

Specifically, the **ambitionMode** flag enables the inclusion in the source code of the stub of facilities used for the emulation of the functional behavior in Jambition. Listing 1 shows the definition of the Service State Machine (SSM) (line 4), the simulator that browses the SSM in order to emulates the correct functional behavior (line 9), and a private utility method (lines 11-30).

```
10    /*
11     * The SSM object.
12     */
13    private info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine aMbItIoNssm;
14
15    /*
16     * The simulator is generated
17     */
18    private info.frantzen.testing.ssmsimulator.SSMSimulator aMbItIoNsim;
19
20    private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMessage(
21        info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine ssm,
22        info.frantzen.testing.ssmsimulator.ssm.MessageKind kind,
23        info.frantzen.testing.ssmsimulator.ssm.Operation op)
24        throws Exception {
25      java.util.HashSet<info.frantzen.testing.ssmsimulator.ssm.Message> messages = ssm
26          .getMessages();
27      for (java.util.Iterator it = messages.iterator(); it.hasNext();) {
28        info.frantzen.testing.ssmsimulator.ssm.Message m = (info.frantzen.testing.ssmsimulator.ssm.Message) it
29            .next();
30        if (m.getKind() != info.frantzen.testing.ssmsimulator.ssm.MessageKind.UNOBSERVABLE) {
31          if ((m.getKind() == kind) && (m.getOperation().equals(op))) {
32            return m;
33          }
34        }
35      }
36      throw new Exception(
37          "Cannot find the input SSM message belonging to the Operation "
38              + op.getName() + "!");
39    }
```

Listing 1: Attributes and Local operation included in the code

For each stub, the body of a default constructor is generated. In addition those line required in order to instantiate and to initialize both the simulator, and the SSM object are generated. Please note that once the stub is generated, some parameters required by the simulator have to be manually set by the user. Referring to Listing 2:

- at line 49 set the URL of the WSDL the Web Service that is going to be emulated exports

- at line 53 set the name of the Web Service that is going to be emulated as reported on the WSDL

- at line 57 set the port of the Web Service that is going to be emulated as reported on the WSDL

- at line 61 set the URL of the file with the specification of the SSM

- at line 76 set the URL of the treeSolver that the simulator uses in order to generate meaningful functional values

- at line 77 set the port of the treeSolver that the simulator uses in order to generate meaningful functional values

```
42      /*
43       * To initialise the Simulator, the following items are needed:
44       */
45
46      /*
47       * The URL of the WSDL file
48       */
49      java.net.URL aMbItIoNWSDLUrl = new java.net.URL("Put here the URL of the Service's WSDL");
50      /*
51       * The name of the WSDL-Service
52       */
53      String aMbItIoNservice = "Put here the name of the Service as in the WSDL";
54      /*
55       * The name of the WSDL-Port
56       */
57      String aMbItIoNport = "Put here the port of the Service as in the WSDL";
58      /*
59       * The URL of the SSM Schema Instance
60       */
61      java.net.URL aMbItIoNSSMUrl = new java.net.URL("Put here the URL of the SSM Schema Instance");
62
63      /*
64       * Now we can generate the SSM object. To do so, we use Zsolt's "Minerva" library
65       */
66      aMbItIoNssm = hu.soft4d.jessi.ssm.SSMHandler.generateSSM(
67          aMbItIoNWSDLUrl, aMbItIoNSSMUrl, aMbItIoNservice, aMbItIoNport);
68      /*
69       * Before we can use the SSM in the simulator, the parsers have to be attached
70       * to the switches
71       */
72      aMbItIoNssm.attachParsersToSwitches();
73      /*
74       * Next we generate the socket to the treeSolver.
75       */
76      String aMbItIoNsolverHost = "Put here the URL of the Solver";
77      int aMbItIoNsolverPort = "Put here the Port the Solver";
78      java.net.Socket aMbItIoNsolverSocket = new java.net.Socket(
79          aMbItIoNsolverHost, aMbItIoNsolverPort);
80      /*
81       * The treeSolver sends a welcome message, we remove it from the stream
82       */
83      new java.io.BufferedReader(new java.io.InputStreamReader(
84          aMbItIoNsolverSocket.getInputStream())).readLine();
85      /*
86       * The simulator can use an external tool to display sequence diagrams
87       * of the messages exchanged. // I skip this here since this takes extra
88       * ressources/
89       */
90
91      /*
92       * The simulator needs a logger to log to
93       */
94      java.util.logging.Logger aMbItIoNlogger = java.util.logging.Logger
95          .getLogger("");
96
97      /* The simulator is generated */
98
99      aMbItIoNsim = new info.frantzen.testing.ssmsimulator.SSMSimulator(
100         aMbItIoNssm, aMbItIoNsolverSocket, aMbItIoNlogger);
101
102     /*
103      * If Double variables are used we assume this models money
104      * (experimental). In any case, do this:
105      */
106     info.frantzen.testing.ssmsimulator.types.ST_PseudoPosDouble.postPointLength = 2;
107     /*
108      * Now the Simulator is ready.
109      * -----------------------------------------------------------------------
110      */
```

Listing 2: Code Included Into the Default Class Constructor

For each operation exported by the Web Service, PUPPET include in the correspondent method body the code emulating the functional behavior. Listing 3 shows the code line instantiating the local variables used in order to interact with the simulator. Note that both the name and the type of the operation match with the parameters generated at line 279.

```
272    public void orderShipment(int ref, services.Address adr) throws java.rmi.RemoteException {
273        long aMbItIoNinvocationTime = 0;
274        try {
275            aMbItIoNinvocationTime = System.currentTimeMillis();
276            /*
277             * Code Generated for Integration with Ambition
278             */
279            info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
                   ssmsimulator.ssm.Operation("orderShipment", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
                   ONEWAY);
280            info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage( aMbItIoNssm,
                   info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
281            info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
                   ssmsimulator.ssm.Valuation();
282            java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
                   aMbItIoNmessage.getType();
283            java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
284            info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
```

Listing 3: Local Variables Configuration

Listings 4 shows an example of the set of lines generated for each parameter that any operation exports.

```
286            /*
287             * Generated Parameter 0
288             */
289            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
290            info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
                   ssmsimulator.types.ST_PosIntInstance(ref);
291            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), refInstance);
292
293            /*
294             * Generated Parameter 1
295             */
296            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
297            Object[] aMbItIoNparameterValues = new Object[2];
298            aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                   getFirstName());
299            aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                   getLastName());
300            info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
                   ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
301            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),adrInstance);
```

Listing 4: The Generation of the Operation's Parameters

In the end, the last piece of code that Listing 5 shows concerns the interrogation to the functional simulator. Note that here the simulator can potentially spot a failure, namely when this message is not specified in the SSM. Here the generated stub is thus able to do also functional testing.

```
303            /*
304             * The valuation is ready, we can construct an instantiated message
305             */
306            info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
                   ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);
307
308            /*
309             * This instantiated message can now be given to the simulator. Note
310             * that here the simulator can potentially spot a failure, namely
311             * when this message is not specified in the SSM! In that sense,
312             * here we do testing.
313             */
314            aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
315        } catch (Exception genericException) {
316            throw new java.rmi.RemoteException(genericException.getMessage());
317        }
```

Listing 5: The Generation of the Operation's Parameters

In case the operation has to return a meaningful value, an additional set of code lines is added to the body of the operation. Specifically, when the simulator knows the input, it is possible to query it for a correct response. First we ask the simulator for all currently activated output transitions (line 158 at Listing 6). Out of all possible output switches, we randomly choose one and check if it has a solution. If yes, we take it. If not, we choose randomly the next one (lines 165-180 at Listing 6)).

```
153        /*
154         * Ok, the simulator knows the input. Now we need a functionally
155         * correct response to this call. We first ask the simulator for all
156         * currently activated output transitions.
157         */
158        java.util.ArrayList aMbItIoNoutputs = new java.util.ArrayList(aMbItIoNsim.getCurrentOutputSwitches());
159
160        /*
161         * Out of all possible output switches, we randomly choose one and
162         * check if it has a solution. If yes, we take it. If not, we choose
163         * randomly the next one.
164         */
165        boolean aMbItIoNnoSolutionFound = true;
166        info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNnextOutput = null;
167        java.util.Random aMbItIoNrandom = new java.util.Random();
168        while (!aMbItIoNoutputs.isEmpty() && aMbItIoNnoSolutionFound) {
169            info.frantzen.testing.ssmsimulator.ssm.Switch aMbItIoNcandidate = (info.frantzen.testing.
                 ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));
170
171            /*
172             * try to find a solution, if yes, fine, if not, remove the
173             * candidate
174             */
175            aMbItIoNnextOutput = aMbItIoNsim.findSolution(aMbItIoNcandidate);
176            if (aMbItIoNnextOutput == null)
177                aMbItIoNoutputs.remove(aMbItIoNcandidate);
178            else
179                aMbItIoNnoSolutionFound = false;
180        }
181        if (aMbItIoNnextOutput == null)
182            throw new Exception("Failure_in_SSM!_No_output_for_synchronous_input_specified!");
183        /*
184         * Ok, we have now a feasible and functionally correct output:
185         * nextOutput Before we send this output to the Service out there,
186         * we tell so to the simulator:
187         */
188        aMbItIoNsim.processInstantiatedMessageNoBackup(aMbItIoNnextOutput);
189
190        /*
191         * What is left to do, is to map this instantiated message back to a
192         * real returnValue.
193         */
194        info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNreturnMessage = aMbItIoNnextOutput.getMessage()
                 ;
195        String aMbItIoNreturnVarName = ((info.frantzen.testing.ssmsimulator.ssm.InteractionVariable)
                 aMbItIoNreturnMessage.getType().iterator().next()).getName();
196        info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
                 getValuation();
197        info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
                 .getSingleInstance(aMbItIoNreturnVarName);
198        String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
199        aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue()
                 ),aMbItIoNarrayRepresentation[1],(Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()),(
                 Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
200        /*
201         * Now send the returnValue back to the calling service. That's it.
202         */
```

Listing 6: The Generation of the Meaningful Return Value

Once a feasible and functionally correct output is found, before sending it to the calling Service, the simulator has to store the output we choose (line 188 at Listing 6). Thus, what is left to do, is to map the output message back to a real return value (line 194-199 at Listing 6).

For the sake of completeness, in the appendix is reported the Java source code emulating a Warehouse Web Service (see the example in Chapter 4.6). The whole code of the stub was automatically generated by Puppet with the **ambitionMode** enabled.

4.5 Example

In this section, an example scenario on how to describe a WS-Agreement specification for the eHealth application domain is presented. The scenario is inspired to the eHealth scenarios proposed by the PLASTIC industrial partners. We remind that the WS Agreement file could also be automatically generated by means of the SlangMon editor provided by the Plastic Platform (see [5] for details).
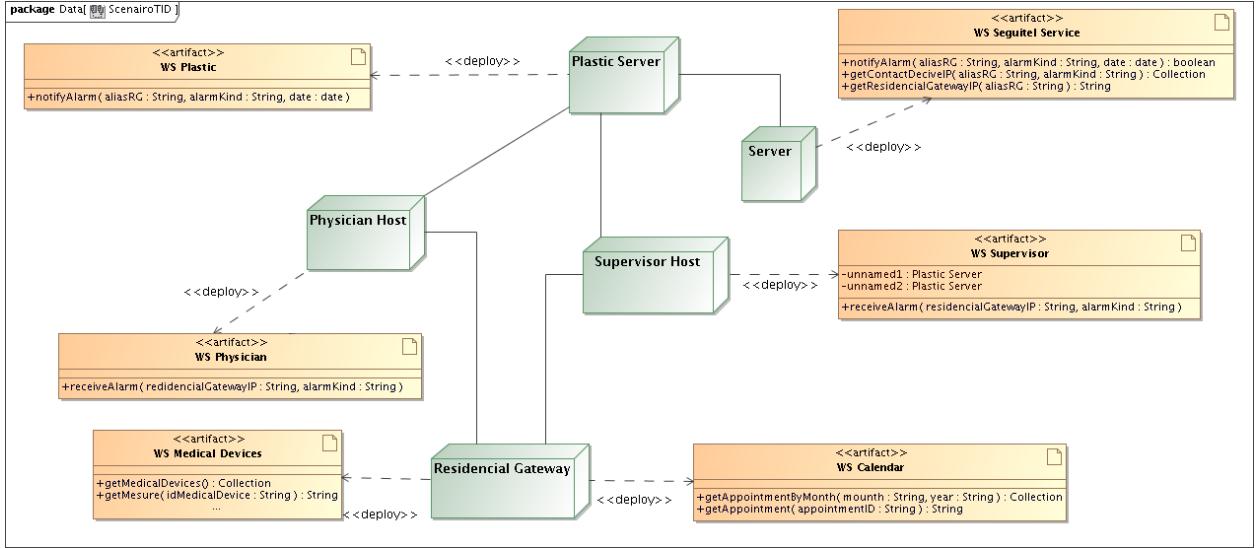
Figure 8: Scenario 3 Deployment Diagram

4.5.1 Scenario Description

The scenario is structured as depicted in Figure 8. Five web services are involved in this example:

WSPlastic : Is the Web Service that interfaces the current eHealth system with the new Plastic environment. In this scenario description we only refer to one of the possible operations and feature that it exports. Specifically the operation **notifyAlarm** is aimed at both collect and process the alarm messages coming from the eHealth part of the application that runs on Plastic. It is invoked when an alarm is raised. It takes as input the name of the Residential Gateway where the alarm comes from, the kind of the alarm and the date when it was raised.

In the scenario, WSPlastic represents the new service that have to be tested. Puppet here is used to automatically build the portion of the system that interacts with the service under test (i.e. WSPlastic)

WSSeguitelService : This Web Service represents the current eHealth application. It exports the following operations: **notifyAlarm**, **getResidentialGatewayIP**, **getContactDeciveIP**. **notifyAlarm**, takes as input the name of the Residential Gateway where the alarm comes from, the kind of the alarm and the date when it was raised. Both the other two exported operations take as input the logic name of a Residential Gateway. Thus **getResidentialGatewayIP** returns the IP address associated with the input label while **getContactDeciveIP** the list of IP addresses that should be contacted in case of alarm.

WSDoctor : It is the service deployed on the device that the doctor uses to interface with the eHealth system. The hosting device could be either a usual wired device as a PC or a mobile and wireless device such as a smart phone. The Web Service exports the **receiveAlarm** operation. Such operation takes as input both the IP address of the Residential Gateway where the alarm was raised and the kind of the alarm.

WSSupervisor : It is a service deployed on the device that a supervisor uses to interface with the eHealth system. The supervisor of a patient is the person that could assist the patient for non critical situation. In those cases that are classified as non critical, some kind of alarm could be forwarded to the supervisor instead of the doctor.

As for the doctor, also here the hosting device could be either a usual wired device as a PC or a mobile and wireless device such as a smart phone. The Web Service exports the **receiveAlarm** operation. Such operation takes as input both the IP address of the Residential Gateway where the alarm was raised and the kind of the alarm.

WSMedicalDevice : Each Residential Gateway controls several hardware medical devices. Each medical device is identified on the Residential Gateway by means of a unique identification code.

This Web Service interfaces the medical devices hosted by the Residential Gateway on the Plastic Network. It exports two operations. **getMedicalDevices** returns a collection of the controlled medical devices. To control an eHealth parameter monitored by a medical device, the web service has to be invoked on the **getMeasure** operation providing the id code of the medical device as input.

WSCalendar : Each Residential Gateway holds the lists of the periodic appointments that a supervisor plans with the patient. This Web Service interfaces the Plastic Network to this feature exporting the methods : **getAppointmentByMonth**, and **getAppointment**. The former gives information on the appointments already scheduled in a given month of a year. The latter is used to create a new one.

In this scenario, the Web Services described above are supposed to be deployed on different and distributed platforms. Specifically, WSPlastic is deployed on a **PlasticServer**, while the WSSeguitelService is supposed to run on the **eHealthServer**. Nevertheless, in principle the two server could be also the same.

4.5.2 Actors Interactions in the Scenario

This section provides a brief description on how the Web Services described in Section 4.5.1 interact. Figure 9 reports a UML Sequence Diagram describing these interactions.

When a alarm is notified to **WSPlastic**, as first step it forwards the event to **WSSequitelService** invoking the **notifyAlarm** method. The Plastic Web Service also invokes **getResidentialGatewayIP** obtaining the IP address of the Residential Gateway where the alarm was raised. Thus, it gets the list of the IP addresses that must be contacted invoking **getContactDeciveIP** on **WSSequitelService**. The obtained list depends on the kind of the received alarm.

Due to the kind of contact list the **WSPlastic** starts to invoke the appropriate Web Service. The Web Services to invoke are supposed to be deployed and reachable by means of the address list. This phase will continue until any confirmation of the handled alarm is obtained either by the doctor or the supervisor.

In the following, the case where the alarm kind is an emergency (i.e. "EMERGENCY") is described. **WSPlastic** extracts an IP address from the address list. Then, it invokes **receiveAlarm** on **WSDoctor** using the IP address as endpoint. If the target Web Service decides to accept the alarm handling **WSPlastic** ends considering the problem solved. On the other hand, if **WSDoctor** does not accept to handle the notification or due to a QoS agreement violation (see Section 5.7), **WSPlastic** proceeds by extracting the next endpoint of the target Web Service.

When the doctor agrees on handle the alarm, he/she contacts **WSMedicalDevices** deployed on the referred Residential Gateway. Then, **WSDoctor** collects the list of the medical devices controlled by the Residential Gateway. The monitoring of the patient parameters is performed invoking the **getMeasure** method on those devices that are considered important for the clinical status.

In the following, the case where the alarm kind is not critical (i.e. "NOT CONFIRMATION"). **WSPlastic** extracts an IP address from the address list. Then, it invokes **receiveAlarm** on **WSSupervisor** using the IP address as endpoint. If the target Web Service decides to accept the alarm handling **WSPlastic** ends considering the problem solved. On the other hand, if **WSSupervisor** does not accept to handle the notification or due to a QoS agreement violation (see Section 5.7), **WSPlastic** proceeds extracting the next endpoint of the target Web Service.

When the supervisor agrees on handle the alarm, he/she contacts **WSCalendar** deployed on the referred Residential Gateway. In the end, the **WSSupervisor** queries the Residential Gateway to schedule an appointment with the patient.

4.5.3 QoS Properties Definition

As described in Section 4.5.1, the Web Services considered in this scenario could be deployed on different machines. In particular, both **WSDoctor** and **WSSupervisor** could be deployed on a usual wired device as a PC or a mobile and wireless device such as a smart phone.

Is it clear that the QoS properties of the service could depend on where the service in actually deployed. For example, if a the Web Service is deployed on a wireless node it is not always given that it is possible to reach it. On the other hand, if a Web Service is deployed on a standard PC it operates at higher performances than one deployed on a smart phone.

Moreover, a method can behave differently depending on the parameters it receives. For example, the processing time of **getMeasure** on **WSMedicalDevice** directly depends on the kind of measurement that is performed and the medical device that is used.
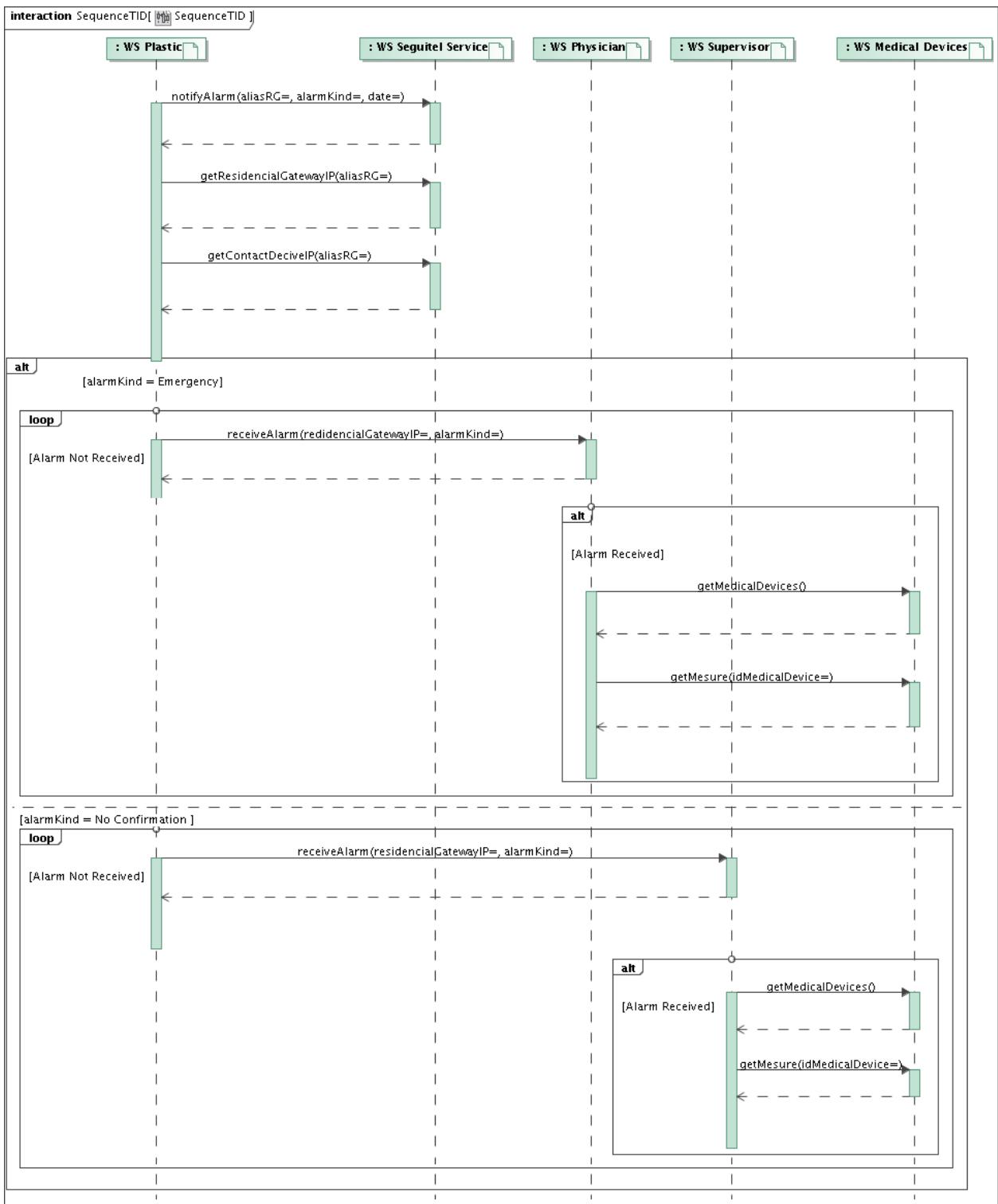
Figure 9: Scenario 3 Sequence Diagram

	Latency (msec)	Reliability		Conditions
WSSeguitelService:getContactDeciveIP	2000			alarmKind="Emergency"
WSSeguitelService:getContactDeciveIP	1000			alarmKind="No Confirmation"
WSDoctor:receiveAlarm	6000	WinSize Max Fails in Win	30000 5	deployedOn="MobileNode"
WSDoctor:receiveAlarm	2000	WinSize Max Fails in Win	30000 1	deployedOn="WiredServer"
WSSupervisor:receiveAlarm	10000	WinSize Max Fails in Win	30000 5	deployedOn="MobileNode"
WSSupervisor:receiveAlarm	6000	WinSize Max Fails in Win	30000 1	deployedOn="WiredServer"
WSMedicalDevice:getMeasure	3000			idMedicalDevice="device_1"
WSMedicalDevice:getMeasure	10000			idMedicalDevice="device_2"

Table 2: QoS Properties

The QoS levels admitted in the scenario here considered are formalized in an agreement. Table 2[3] reports a short version of the extra functional properties that are supposed to be respected in the scenario. Time are given in milliseconds. In the appendix below, the listings reporting the complete version of the XML document expressing the agreement are reported.

4.6 Reference To Jambition

For any reference to Jambition in this document, please refer to [4].

5 Appendix

5.1 Abstract WSDL: WSDoctor

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSDoctor" xmlns:apachesoap="http://xml.
       apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSDoctor" xmlns:intf="http://
       localhost:8080/axis/services/WSDoctor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl
       ="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http:
       //www.w3.org/2001/XMLSchema">
3  <!--WSDL created by Apache Axis version: 1.4
4  Built on Apr 22, 2006 (06:55:48 PDT)-->
5
6    <wsdl:message name="receiveAlarmRequest">
7
8      <wsdl:part name="residencialGatewayIP" type="soapenc:string"/>
9
10     <wsdl:part name="alarmGender" type="soapenc:string"/>
11
12   </wsdl:message>
13
14   <wsdl:message name="receiveAlarmResponse">
15
16   </wsdl:message>
17
18   <wsdl:portType name="WSDoctor">
19
20     <wsdl:operation name="receiveAlarm" parameterOrder="residencialGatewayIP_alarmGender">
21
22       <wsdl:input message="impl:receiveAlarmRequest" name="receiveAlarmRequest"/>
23
24       <wsdl:output message="impl:receiveAlarmResponse" name="receiveAlarmResponse"/>
25
26     </wsdl:operation>
27
28   </wsdl:portType>
29
30   <wsdl:binding name="WSDoctorSoapBinding" type="impl:WSDoctor">
31
32     <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
33
34     <wsdl:operation name="receiveAlarm">
35
36       <wsdlsoap:operation soapAction=""/>
37
```

[3]The values in this table has to be considered just as an example.

```
38        <wsdl:input name="receiveAlarmRequest">
39
40          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                  use="encoded"/>
41
42        </wsdl:input>
43
44        <wsdl:output name="receiveAlarmResponse">
45
46          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                  localhost:8080/axis/services/WSDoctor" use="encoded"/>
47
48        </wsdl:output>
49
50      </wsdl:operation>
51
52    </wsdl:binding>
53
54    <wsdl:service name="WSDoctorService">
55
56      <wsdl:port binding="impl:WSDoctorSoapBinding" name="WSDoctor">
57
58        <wsdlsoap:address location="http://localhost:8080/axis/services/WSDoctor"/>
59
60      </wsdl:port>
61
62    </wsdl:service>
63
64  </wsdl:definitions>
```

5.2   Abstract WSDL: WSCalendar

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSCalendar" xmlns:apachesoap="http://xml
       .apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSCalendar" xmlns:intf="http://
       localhost:8080/axis/services/WSCalendar" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3  <!--WSDL created by Apache Axis version: 1.4
4  Built on Apr 22, 2006 (06:55:48 PDT)-->
5   <wsdl:types>
6    <schema targetNamespace="http://localhost:8080/axis/services/WSCalendar" xmlns="http://www.w3.org/2001/
         XMLSchema">
7    <import namespace="http://xml.apache.org/xml-soap"/>
8    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
9    <complexType name="ArrayOf_xsd_anyType">
10    <complexContent>
11     <restriction base="soapenc:Array">
12      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
13     </restriction>
14    </complexContent>
15   </complexType>
16   </schema>
17   <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
18    <import namespace="http://localhost:8080/axis/services/WSCalendar"/>
19    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
20    <complexType name="Vector">
21     <sequence>
22      <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
23     </sequence>
24    </complexType>
25   </schema>
26   </wsdl:types>
27
28    <wsdl:message name="getAppointmentByMonthRequest">
29
30      <wsdl:part name="month" type="soapenc:string"/>
31
32      <wsdl:part name="year" type="soapenc:string"/>
33
34    </wsdl:message>
35
36    <wsdl:message name="getAppointmentRequest">
37
38      <wsdl:part name="apointmentID" type="soapenc:string"/>
39
40    </wsdl:message>
41
42    <wsdl:message name="getAppointmentByMonthResponse">
43
44      <wsdl:part name="getAppointmentByMonthReturn" type="impl:ArrayOf_xsd_anyType"/>
45
46    </wsdl:message>
```

```
47
48    <wsdl:message name="getAppointmentResponse">
49
50      <wsdl:part name="getAppointmentReturn" type="soapenc:string"/>
51
52    </wsdl:message>
53
54    <wsdl:portType name="WSCalendar">
55
56      <wsdl:operation name="getAppointmentByMonth" parameterOrder="month_year">
57
58        <wsdl:input message="impl:getAppointmentByMonthRequest" name="getAppointmentByMonthRequest"/>
59
60        <wsdl:output message="impl:getAppointmentByMonthResponse" name="getAppointmentByMonthResponse"/>
61
62      </wsdl:operation>
63
64      <wsdl:operation name="getAppointment" parameterOrder="apointmentID">
65
66        <wsdl:input message="impl:getAppointmentRequest" name="getAppointmentRequest"/>
67
68        <wsdl:output message="impl:getAppointmentResponse" name="getAppointmentResponse"/>
69
70      </wsdl:operation>
71
72    </wsdl:portType>
73
74    <wsdl:binding name="WSCalendarSoapBinding" type="impl:WSCalendar">
75
76      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
77
78      <wsdl:operation name="getAppointmentByMonth">
79
80        <wsdlsoap:operation soapAction=""/>
81
82        <wsdl:input name="getAppointmentByMonthRequest">
83
84          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                use="encoded"/>
85
86        </wsdl:input>
87
88        <wsdl:output name="getAppointmentByMonthResponse">
89
90          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSCalendar" use="encoded"/>
91
92        </wsdl:output>
93
94      </wsdl:operation>
95
96      <wsdl:operation name="getAppointment">
97
98        <wsdlsoap:operation soapAction=""/>
99
100       <wsdl:input name="getAppointmentRequest">
101
102         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                use="encoded"/>
103
104       </wsdl:input>
105
106       <wsdl:output name="getAppointmentResponse">
107
108         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSCalendar" use="encoded"/>
109
110       </wsdl:output>
111
112     </wsdl:operation>
113
114   </wsdl:binding>
115
116   <wsdl:service name="WSCalendarService">
117
118     <wsdl:port binding="impl:WSCalendarSoapBinding" name="WSCalendar">
119
120       <wsdlsoap:address location="http://localhost:8080/axis/services/WSCalendar"/>
121
122     </wsdl:port>
123
124   </wsdl:service>
125
126 </wsdl:definitions>
```

5.3 Abstract WSDL: WSSupervisor

```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSSupervisor" xmlns:apachesoap="http://
      xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSupervisor" xmlns:intf="http://
      localhost:8080/axis/services/WSSupervisor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3  <!--WSDL created by Apache Axis version: 1.4
4  Built on Apr 22, 2006 (06:55:48 PDT)-->
5
6    <wsdl:message name="receiveAlarmResponse">
7
8    </wsdl:message>
9
10   <wsdl:message name="receiveAlarmRequest">
11
12     <wsdl:part name="residencialGatewayIP" type="soapenc:string"/>
13
14     <wsdl:part name="alarmGender" type="soapenc:string"/>
15
16   </wsdl:message>
17
18   <wsdl:portType name="WSSupervisor">
19
20     <wsdl:operation name="receiveAlarm" parameterOrder="residencialGatewayIP alarmGender">
21
22       <wsdl:input message="impl:receiveAlarmRequest" name="receiveAlarmRequest"/>
23
24       <wsdl:output message="impl:receiveAlarmResponse" name="receiveAlarmResponse"/>
25
26     </wsdl:operation>
27
28   </wsdl:portType>
29
30   <wsdl:binding name="WSSupervisorSoapBinding" type="impl:WSSupervisor">
31
32     <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
33
34     <wsdl:operation name="receiveAlarm">
35
36       <wsdlsoap:operation soapAction=""/>
37
38       <wsdl:input name="receiveAlarmRequest">
39
40         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
             use="encoded"/>
41
42       </wsdl:input>
43
44       <wsdl:output name="receiveAlarmResponse">
45
46         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSSupervisor" use="encoded"/>
47
48       </wsdl:output>
49
50     </wsdl:operation>
51
52   </wsdl:binding>
53
54   <wsdl:service name="WSSupervisorService">
55
56     <wsdl:port binding="impl:WSSupervisorSoapBinding" name="WSSupervisor">
57
58       <wsdlsoap:address location="http://localhost:8080/axis/services/WSSupervisor"/>
59
60     </wsdl:port>
61
62   </wsdl:service>
63
64  </wsdl:definitions>
```

5.4 Abstract WSDL: WSPlastic

```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSPlastic" xmlns:apachesoap="http://xml.
      apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSPlastic" xmlns:intf="http://
      localhost:8080/axis/services/WSPlastic" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3  <!--WSDL created by Apache Axis version: 1.4
```

```
4     Built on Apr 22, 2006 (06:55:48 PDT)-->
5
6       <wsdl:message name="notifyAlarmRequest">
7
8         <wsdl:part name="aliasRG" type="soapenc:string"/>
9
10        <wsdl:part name="alarmGender" type="soapenc:string"/>
11
12        <wsdl:part name="date" type="xsd:dateTime"/>
13
14      </wsdl:message>
15
16      <wsdl:message name="notifyAlarmResponse">
17
18      </wsdl:message>
19
20      <wsdl:portType name="WSPlastic">
21
22        <wsdl:operation name="notifyAlarm" parameterOrder="aliasRG alarmGender date">
23
24          <wsdl:input message="impl:notifyAlarmRequest" name="notifyAlarmRequest"/>
25
26          <wsdl:output message="impl:notifyAlarmResponse" name="notifyAlarmResponse"/>
27
28        </wsdl:operation>
29
30      </wsdl:portType>
31
32      <wsdl:binding name="WSPlasticSoapBinding" type="impl:WSPlastic">
33
34        <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
35
36        <wsdl:operation name="notifyAlarm">
37
38          <wsdlsoap:operation soapAction=""/>
39
40          <wsdl:input name="notifyAlarmRequest">
41
42            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                     use="encoded"/>
43
44          </wsdl:input>
45
46          <wsdl:output name="notifyAlarmResponse">
47
48            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                     localhost:8080/axis/services/WSPlastic" use="encoded"/>
49
50          </wsdl:output>
51
52        </wsdl:operation>
53
54      </wsdl:binding>
55
56      <wsdl:service name="WSPlasticService">
57
58        <wsdl:port binding="impl:WSPlasticSoapBinding" name="WSPlastic">
59
60          <wsdlsoap:address location="http://localhost:8080/axis/services/WSPlastic"/>
61
62        </wsdl:port>
63
64      </wsdl:service>
65
66    </wsdl:definitions>
```

5.5 Abstract WSDL: WSSeguitelService

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSSeguitelService" xmlns:apachesoap="
          http://xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSeguitelService"
          xmlns:intf="http://localhost:8080/axis/services/WSSeguitelService" xmlns:soapenc="http://schemas.xmlsoap.
          org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.
          org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3     <!--WSDL created by Apache Axis version: 1.4
4     Built on Apr 22, 2006 (06:55:48 PDT)-->
5      <wsdl:types>
6       <schema targetNamespace="http://localhost:8080/axis/services/WSSeguitelService" xmlns="http://www.w3.org/2001/
            XMLSchema">
7        <import namespace="http://xml.apache.org/xml-soap"/>
8        <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
9        <complexType name="ArrayOf_xsd_anyType">
10        <complexContent>
```

```
11      <restriction base="soapenc:Array">
12       <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
13      </restriction>
14     </complexContent>
15    </complexType>
16   </schema>
17   <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
18    <import namespace="http://localhost:8080/axis/services/WSSeguitelService"/>
19    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
20    <complexType name="Vector">
21     <sequence>
22      <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
23     </sequence>
24    </complexType>
25   </schema>
26  </wsdl:types>

28   <wsdl:message name="getResidencialGatewayIPResponse">

30     <wsdl:part name="getResidencialGatewayIPReturn" type="soapenc:string"/>

32   </wsdl:message>

34   <wsdl:message name="getResidencialGatewayIPRequest">

36     <wsdl:part name="aliasRG" type="soapenc:string"/>

38   </wsdl:message>

40   <wsdl:message name="notifyAlarmRequest">

42     <wsdl:part name="aliasRG" type="soapenc:string"/>

44     <wsdl:part name="alarmGender" type="soapenc:string"/>

46     <wsdl:part name="date" type="xsd:dateTime"/>

48   </wsdl:message>

50   <wsdl:message name="getConnectedDeviceIPRequest">

52     <wsdl:part name="aliasRG" type="soapenc:string"/>

54     <wsdl:part name="alarmGender" type="soapenc:string"/>

56   </wsdl:message>

58   <wsdl:message name="notifyAlarmResponse">

60     <wsdl:part name="notifyAlarmReturn" type="xsd:boolean"/>

62   </wsdl:message>

64   <wsdl:message name="getConnectedDeviceIPResponse">

66     <wsdl:part name="getConnectedDeviceIPReturn" type="impl:ArrayOf_xsd_anyType"/>

68   </wsdl:message>

70   <wsdl:portType name="WSSeguitelService">

72     <wsdl:operation name="notifyAlarm" parameterOrder="aliasRG␣alarmGender␣date">

74       <wsdl:input message="impl:notifyAlarmRequest" name="notifyAlarmRequest"/>

76       <wsdl:output message="impl:notifyAlarmResponse" name="notifyAlarmResponse"/>

78     </wsdl:operation>

80     <wsdl:operation name="getConnectedDeviceIP" parameterOrder="aliasRG␣alarmGender">

82       <wsdl:input message="impl:getConnectedDeviceIPRequest" name="getConnectedDeviceIPRequest"/>

84       <wsdl:output message="impl:getConnectedDeviceIPResponse" name="getConnectedDeviceIPResponse"/>

86     </wsdl:operation>

88     <wsdl:operation name="getResidencialGatewayIP" parameterOrder="aliasRG">

90       <wsdl:input message="impl:getResidencialGatewayIPRequest" name="getResidencialGatewayIPRequest"/>

92       <wsdl:output message="impl:getResidencialGatewayIPResponse" name="getResidencialGatewayIPResponse"/>

94     </wsdl:operation>
```

21

```wsdl
  95
  96       </wsdl:portType>
  97
  98       <wsdl:binding name="WSSeguitelServiceSoapBinding" type="impl:WSSeguitelService">
  99
 100          <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 101
 102          <wsdl:operation name="notifyAlarm">
 103
 104             <wsdlsoap:operation soapAction=""/>
 105
 106             <wsdl:input name="notifyAlarmRequest">
 107
 108                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                        use="encoded"/>
 109
 110             </wsdl:input>
 111
 112             <wsdl:output name="notifyAlarmResponse">
 113
 114                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                        localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
 115
 116             </wsdl:output>
 117
 118          </wsdl:operation>
 119
 120          <wsdl:operation name="getConnectedDeviceIP">
 121
 122             <wsdlsoap:operation soapAction=""/>
 123
 124             <wsdl:input name="getConnectedDeviceIPRequest">
 125
 126                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                        use="encoded"/>
 127
 128             </wsdl:input>
 129
 130             <wsdl:output name="getConnectedDeviceIPResponse">
 131
 132                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                        localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
 133
 134             </wsdl:output>
 135
 136          </wsdl:operation>
 137
 138          <wsdl:operation name="getResidencialGatewayIP">
 139
 140             <wsdlsoap:operation soapAction=""/>
 141
 142             <wsdl:input name="getResidencialGatewayIPRequest">
 143
 144                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                        use="encoded"/>
 145
 146             </wsdl:input>
 147
 148             <wsdl:output name="getResidencialGatewayIPResponse">
 149
 150                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                        localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
 151
 152             </wsdl:output>
 153
 154          </wsdl:operation>
 155
 156       </wsdl:binding>
 157
 158       <wsdl:service name="WSSeguitelServiceService">
 159
 160          <wsdl:port binding="impl:WSSeguitelServiceSoapBinding" name="WSSeguitelService">
 161
 162             <wsdlsoap:address location="http://localhost:8080/axis/services/WSSeguitelService"/>
 163
 164          </wsdl:port>
 165
 166       </wsdl:service>
 167
 168    </wsdl:definitions>
```

5.6 Abstract WSDL: WSMedicalDevice

```xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSMedicalDevice" xmlns:apachesoap="http:
        //xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSMedicalDevice" xmlns:intf="
        http://localhost:8080/axis/services/WSMedicalDevice" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
        encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/
        soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <!--WSDL created by Apache Axis version: 1.4
4   Built on Apr 22, 2006 (06:55:48 PDT)-->
5    <wsdl:types>
6     <schema targetNamespace="http://localhost:8080/axis/services/WSMedicalDevice" xmlns="http://www.w3.org/2001/
        XMLSchema">
7      <import namespace="http://xml.apache.org/xml-soap"/>
8      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
9      <complexType name="ArrayOf_xsd_anyType">
10      <complexContent>
11       <restriction base="soapenc:Array">
12        <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
13       </restriction>
14      </complexContent>
15     </complexType>
16    </schema>
17    <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
18     <import namespace="http://localhost:8080/axis/services/WSMedicalDevice"/>
19     <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
20     <complexType name="Vector">
21      <sequence>
22       <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
23      </sequence>
24     </complexType>
25    </schema>
26   </wsdl:types>
27
28     <wsdl:message name="getMedicalDevicesRequest">
29
30     </wsdl:message>
31
32     <wsdl:message name="getMedicalDevicesResponse">
33
34        <wsdl:part name="getMedicalDevicesReturn" type="impl:ArrayOf_xsd_anyType"/>
35
36     </wsdl:message>
37
38     <wsdl:message name="getMeasureRequest">
39
40        <wsdl:part name="idMedicalDevice" type="soapenc:string"/>
41
42     </wsdl:message>
43
44     <wsdl:message name="getMeasureResponse">
45
46        <wsdl:part name="getMeasureReturn" type="soapenc:string"/>
47
48     </wsdl:message>
49
50     <wsdl:portType name="WSMedicalDevice">
51
52        <wsdl:operation name="getMedicalDevices">
53
54           <wsdl:input message="impl:getMedicalDevicesRequest" name="getMedicalDevicesRequest"/>
55
56           <wsdl:output message="impl:getMedicalDevicesResponse" name="getMedicalDevicesResponse"/>
57
58        </wsdl:operation>
59
60        <wsdl:operation name="getMeasure" parameterOrder="idMedicalDevice">
61
62           <wsdl:input message="impl:getMeasureRequest" name="getMeasureRequest"/>
63
64           <wsdl:output message="impl:getMeasureResponse" name="getMeasureResponse"/>
65
66        </wsdl:operation>
67
68     </wsdl:portType>
69
70     <wsdl:binding name="WSMedicalDeviceSoapBinding" type="impl:WSMedicalDevice">
71
72        <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
73
74        <wsdl:operation name="getMedicalDevices">
75
76           <wsdlsoap:operation soapAction=""/>
77
78           <wsdl:input name="getMedicalDevicesRequest">
79
```

```
80        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                use="encoded"/>
81
82      </wsdl:input>
83
84      <wsdl:output name="getMedicalDevicesResponse">
85
86        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSMedicalDevice" use="encoded"/>
87
88      </wsdl:output>
89
90    </wsdl:operation>
91
92    <wsdl:operation name="getMeasure">
93
94      <wsdlsoap:operation soapAction=""/>
95
96      <wsdl:input name="getMeasureRequest">
97
98        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                use="encoded"/>
99
100     </wsdl:input>
101
102     <wsdl:output name="getMeasureResponse">
103
104       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSMedicalDevice" use="encoded"/>
105
106     </wsdl:output>
107
108   </wsdl:operation>
109
110  </wsdl:binding>
111
112  <wsdl:service name="WSMedicalDeviceService">
113
114    <wsdl:port binding="impl:WSMedicalDeviceSoapBinding" name="WSMedicalDevice">
115
116      <wsdlsoap:address location="http://localhost:8080/axis/services/WSMedicalDevice"/>
117
118    </wsdl:port>
119
120  </wsdl:service>
121
122 </wsdl:definitions>
```

5.7 WS Agreement

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsag:AgreementOffer xsi:schemaLocation="http://www.ggf.org/namespaces/ws-agreement"
3                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
5                   xmlns:wsag="http://schemas.ggf.org/graap/2005/09/ws-agreement"
6                   xmlns:puppetScope="http://setest0.isti.cnr.it/puppetScope"
7                   xmlns:puppetSLO="http://setest0.isti.cnr.it/puppetSLO"
8                   xmlns:puppetQC="http://setest0.isti.cnr.it/puppetQC"
9                   xmlns:ns="http://setest0.isti.cnr.it/puppet">
10   <wsag:Name>Telefonica_Example_Scenario_3</wsag:Name>
11
12   <wsag:Context />
13
14   <wsag:Terms>
15     <wsag:All>
16       <wsag:GuaranteeTerm wsag:Name="ContactDeciveIP-Term1"
17        wsag:Obligated="ServiceProvider">
18        <wsag:ServiceScope wsag:ServiceName="WSSeguitelService">
19          <puppetScope:PuppetScope>
20           <puppetScope:Method>
21 <!--           <NameMethod>getContactDeciveIP</NameMethod> -->
22            <NameMethod>getConnectedDeviceIP</NameMethod>
23           </puppetScope:Method>
24          </puppetScope:PuppetScope>
25        </wsag:ServiceScope>
26
27        <wsag:QualifyingCondition>
28         <puppetQC:PuppetQC>
29          <puppetQC:StringBinaryExpression>
30           <puppetQC:StringType>
31            <Variable>alarmGender</Variable>
32           </puppetQC:StringType>
33
```

```xml
            <op>equal</op>

            <puppetQC:StringType>
              <Value>Emergency</Value>
            </puppetQC:StringType>
          </puppetQC:StringBinaryExpression>
        </puppetQC:PuppetQC>
      </wsag:QualifyingCondition>

      <wsag:ServiceLevelObjective>
        <puppetSLO:PuppetSLO>
          <puppetSLO:Latency>
            <value>2000</value>

            <puppetSLO:Distribution>
              <Gaussian>10</Gaussian>
            </puppetSLO:Distribution>
          </puppetSLO:Latency>
        </puppetSLO:PuppetSLO>
      </wsag:ServiceLevelObjective>

      <wsag:BusinessValueList>
        <wsag:Penalty>
          <wsag:AssessmentInterval>
            <wsag:Count />
          </wsag:AssessmentInterval>

          <wsag:ValueExpression> 2 </wsag:ValueExpression>
        </wsag:Penalty>
      </wsag:BusinessValueList>
    </wsag:GuaranteeTerm>
    <wsag:GuaranteeTerm wsag:Name="ContactDeciveIP-Term2"
      wsag:Obligated="ServiceProvider">
      <wsag:ServiceScope wsag:ServiceName="WSSeguitelService">
        <puppetScope:PuppetScope>
          <puppetScope:Method>
<!--           <NameMethod>getContactDeciveIP</NameMethod> -->
          <NameMethod>getConnectedDeviceIP</NameMethod>
          </puppetScope:Method>
        </puppetScope:PuppetScope>
      </wsag:ServiceScope>

      <wsag:QualifyingCondition>
        <puppetQC:PuppetQC>
          <puppetQC:StringBinaryExpression>
            <puppetQC:StringType>
              <Variable>alarmGender</Variable>
            </puppetQC:StringType>

            <op>equal</op>

            <puppetQC:StringType>
              <Value>No Confirmation</Value>
            </puppetQC:StringType>
          </puppetQC:StringBinaryExpression>
        </puppetQC:PuppetQC>
      </wsag:QualifyingCondition>

      <wsag:ServiceLevelObjective>
        <puppetSLO:PuppetSLO>
          <puppetSLO:Latency>
            <value>1000</value>

            <puppetSLO:Distribution>
              <Gaussian>10</Gaussian>
            </puppetSLO:Distribution>
          </puppetSLO:Latency>
        </puppetSLO:PuppetSLO>
      </wsag:ServiceLevelObjective>

      <wsag:BusinessValueList>
        <wsag:Penalty>
          <wsag:AssessmentInterval>
            <wsag:Count />
          </wsag:AssessmentInterval>

          <wsag:ValueExpression> 2 </wsag:ValueExpression>
        </wsag:Penalty>
      </wsag:BusinessValueList>
    </wsag:GuaranteeTerm>
    <wsag:ExactlyOne>
      <wsag:GuaranteeTerm wsag:Name="AlarmDoctor-Term1"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="WSDoctor">
```

```
118          <puppetScope:PuppetScope>
119           <puppetScope:Method>
120            <NameMethod>receiveAlarm</NameMethod>
121           </puppetScope:Method>
122          </puppetScope:PuppetScope>
123         </wsag:ServiceScope>
124
125         <wsag:QualifyingCondition>
126          <puppetQC:PuppetQC>
127           <puppetQC:StringBinaryExpression>
128            <puppetQC:StringType>
129             <Variable>deployedOn</Variable>
130            </puppetQC:StringType>
131
132            <op>equal</op>
133
134            <puppetQC:StringType>
135             <Value>MobileNode</Value>
136            </puppetQC:StringType>
137           </puppetQC:StringBinaryExpression>
138          </puppetQC:PuppetQC>
139         </wsag:QualifyingCondition>
140
141         <wsag:ServiceLevelObjective>
142          <puppetSLO:PuppetSLO>
143           <puppetSLO:Latency>
144            <value>6000</value>
145            <puppetSLO:Distribution>
146              <Gaussian>10</Gaussian>
147            </puppetSLO:Distribution>
148           </puppetSLO:Latency>
149           <puppetSLO:Reliability>
150            <Reliabilitywindow>30000</Reliabilitywindow>
151            <ReliabilityPerc>5</ReliabilityPerc>
152            <puppetSLO:Distribution>
153             <Gaussian>100</Gaussian>
154            </puppetSLO:Distribution>
155           </puppetSLO:Reliability>
156          </puppetSLO:PuppetSLO>
157         </wsag:ServiceLevelObjective>
158
159         <wsag:BusinessValueList>
160          <wsag:Penalty>
161           <wsag:AssessmentInterval>
162            <wsag:Count />
163           </wsag:AssessmentInterval>
164
165           <wsag:ValueExpression> 2 </wsag:ValueExpression>
166          </wsag:Penalty>
167         </wsag:BusinessValueList>
168        </wsag:GuaranteeTerm>
169       <wsag:GuaranteeTerm wsag:Name="AlarmDoctor-Term2"
170        wsag:Obligated="ServiceProvider">
171        <wsag:ServiceScope wsag:ServiceName="WSDoctor">
172         <puppetScope:PuppetScope>
173          <puppetScope:Method>
174           <NameMethod>receiveAlarm</NameMethod>
175          </puppetScope:Method>
176         </puppetScope:PuppetScope>
177        </wsag:ServiceScope>
178
179        <wsag:QualifyingCondition>
180         <puppetQC:PuppetQC>
181          <puppetQC:StringBinaryExpression>
182           <puppetQC:StringType>
183            <Variable>deployedOn</Variable>
184           </puppetQC:StringType>
185
186           <op>equal</op>
187
188           <puppetQC:StringType>
189            <Value>WiredServer</Value>
190           </puppetQC:StringType>
191          </puppetQC:StringBinaryExpression>
192         </puppetQC:PuppetQC>
193        </wsag:QualifyingCondition>
194
195        <wsag:ServiceLevelObjective>
196         <puppetSLO:PuppetSLO>
197          <puppetSLO:Latency>
198           <value>2000</value>
199            <puppetSLO:Distribution>
200              <Gaussian>10</Gaussian>
201              </puppetSLO:Distribution>
```

```
202        </puppetSLO:Latency>
203        <puppetSLO:Reliability>
204         <Reliabilitywindow>30000</Reliabilitywindow>
205
206         <ReliabilityPerc>1</ReliabilityPerc>
207
208         <puppetSLO:Distribution>
209          <Gaussian>100</Gaussian>
210         </puppetSLO:Distribution>
211        </puppetSLO:Reliability>
212       </puppetSLO:PuppetSLO>
213      </wsag:ServiceLevelObjective>
214
215      <wsag:BusinessValueList>
216       <wsag:Penalty>
217        <wsag:AssessmentInterval>
218         <wsag:Count />
219        </wsag:AssessmentInterval>
220
221        <wsag:ValueExpression> 2 </wsag:ValueExpression>
222       </wsag:Penalty>
223      </wsag:BusinessValueList>
224     </wsag:GuaranteeTerm>
225    </wsag:ExactlyOne>
226    <wsag:ExactlyOne>
227     <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term1"
228      wsag:Obligated="ServiceProvider">
229      <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
230       <puppetScope:PuppetScope>
231        <puppetScope:Method>
232         <NameMethod>receiveAlarm</NameMethod>
233        </puppetScope:Method>
234       </puppetScope:PuppetScope>
235      </wsag:ServiceScope>
236
237      <wsag:QualifyingCondition>
238       <puppetQC:PuppetQC>
239        <puppetQC:StringBinaryExpression>
240         <puppetQC:StringType>
241          <Variable>deployedOn</Variable>
242         </puppetQC:StringType>
243
244         <op>equal</op>
245
246         <puppetQC:StringType>
247          <Value>MobileNode</Value>
248         </puppetQC:StringType>
249        </puppetQC:StringBinaryExpression>
250       </puppetQC:PuppetQC>
251      </wsag:QualifyingCondition>
252
253      <wsag:ServiceLevelObjective>
254       <puppetSLO:PuppetSLO>
255        <puppetSLO:Latency>
256         <value>10000</value>
257
258         <puppetSLO:Distribution>
259          <Gaussian>10</Gaussian>
260         </puppetSLO:Distribution>
261        </puppetSLO:Latency>
262        <puppetSLO:Reliability>
263         <Reliabilitywindow>30000</Reliabilitywindow>
264
265         <ReliabilityPerc>5</ReliabilityPerc>
266
267         <puppetSLO:Distribution>
268          <Gaussian>100</Gaussian>
269         </puppetSLO:Distribution>
270        </puppetSLO:Reliability>
271       </puppetSLO:PuppetSLO>
272      </wsag:ServiceLevelObjective>
273
274      <wsag:BusinessValueList>
275       <wsag:Penalty>
276        <wsag:AssessmentInterval>
277         <wsag:Count />
278        </wsag:AssessmentInterval>
279
280        <wsag:ValueExpression> 2 </wsag:ValueExpression>
281       </wsag:Penalty>
282      </wsag:BusinessValueList>
283     </wsag:GuaranteeTerm>
284     <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term2"
285      wsag:Obligated="ServiceProvider">
```

```
286          <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
287           <puppetScope:PuppetScope>
288            <puppetScope:Method>
289             <NameMethod>receiveAlarm</NameMethod>
290            </puppetScope:Method>
291           </puppetScope:PuppetScope>
292          </wsag:ServiceScope>
293
294          <wsag:QualifyingCondition>
295           <puppetQC:PuppetQC>
296            <puppetQC:StringBinaryExpression>
297             <puppetQC:StringType>
298              <Variable>deployedOn</Variable>
299             </puppetQC:StringType>
300
301             <op>equal</op>
302
303             <puppetQC:StringType>
304              <Value>WiredServer</Value>
305             </puppetQC:StringType>
306            </puppetQC:StringBinaryExpression>
307           </puppetQC:PuppetQC>
308          </wsag:QualifyingCondition>
309
310          <wsag:ServiceLevelObjective>
311           <puppetSLO:PuppetSLO>
312            <puppetSLO:Latency>
313             <value>6000</value>
314
315              <puppetSLO:Distribution>
316               <Gaussian>10</Gaussian>
317              </puppetSLO:Distribution>
318            </puppetSLO:Latency>
319            <puppetSLO:Reliability>
320             <Reliabilitywindow>30000</Reliabilitywindow>
321
322             <ReliabilityPerc>1</ReliabilityPerc>
323
324             <puppetSLO:Distribution>
325              <Gaussian>100</Gaussian>
326             </puppetSLO:Distribution>
327            </puppetSLO:Reliability>
328           </puppetSLO:PuppetSLO>
329          </wsag:ServiceLevelObjective>
330
331          <wsag:BusinessValueList>
332           <wsag:Penalty>
333            <wsag:AssessmentInterval>
334             <wsag:Count />
335            </wsag:AssessmentInterval>
336
337            <wsag:ValueExpression> 2 </wsag:ValueExpression>
338           </wsag:Penalty>
339          </wsag:BusinessValueList>
340         </wsag:GuaranteeTerm>
341        </wsag:ExactlyOne>
342        <wsag:GuaranteeTerm wsag:Name="MedicalDevice-Term1"
343         wsag:Obligated="ServiceProvider">
344         <wsag:ServiceScope wsag:ServiceName="WSMedicalDevice">
345          <puppetScope:PuppetScope>
346           <puppetScope:Method>
347            <NameMethod>getMeasure</NameMethod>
348           </puppetScope:Method>
349          </puppetScope:PuppetScope>
350         </wsag:ServiceScope>
351
352         <wsag:QualifyingCondition>
353          <puppetQC:PuppetQC>
354           <puppetQC:StringBinaryExpression>
355            <puppetQC:StringType>
356             <Variable>idMedicalDevice</Variable>
357            </puppetQC:StringType>
358
359            <op>equal</op>
360
361            <puppetQC:StringType>
362             <Value>device_1</Value>
363            </puppetQC:StringType>
364           </puppetQC:StringBinaryExpression>
365          </puppetQC:PuppetQC>
366         </wsag:QualifyingCondition>
367
368         <wsag:ServiceLevelObjective>
369          <puppetSLO:PuppetSLO>
```

```
370        <puppetSLO:Latency>
371          <value>3000</value>
372
373          <puppetSLO:Distribution>
374            <Gaussian>10</Gaussian>
375          </puppetSLO:Distribution>
376        </puppetSLO:Latency>
377      </puppetSLO:PuppetSLO>
378    </wsag:ServiceLevelObjective>
379
380    <wsag:BusinessValueList>
381      <wsag:Penalty>
382        <wsag:AssessmentInterval>
383          <wsag:Count />
384        </wsag:AssessmentInterval>
385
386        <wsag:ValueExpression> 2 </wsag:ValueExpression>
387      </wsag:Penalty>
388    </wsag:BusinessValueList>
389  </wsag:GuaranteeTerm>
390  <wsag:GuaranteeTerm wsag:Name="MedicalDevice-Term2"
391    wsag:Obligated="ServiceProvider">
392    <wsag:ServiceScope wsag:ServiceName="WSMedicalDevice">
393      <puppetScope:PuppetScope>
394        <puppetScope:Method>
395          <NameMethod>getMeasure</NameMethod>
396        </puppetScope:Method>
397      </puppetScope:PuppetScope>
398    </wsag:ServiceScope>
399
400    <wsag:QualifyingCondition>
401      <puppetQC:PuppetQC>
402        <puppetQC:StringBinaryExpression>
403          <puppetQC:StringType>
404            <Variable>idMedicalDevice</Variable>
405          </puppetQC:StringType>
406
407          <op>equal</op>
408
409          <puppetQC:StringType>
410            <Value>device_2</Value>
411          </puppetQC:StringType>
412        </puppetQC:StringBinaryExpression>
413      </puppetQC:PuppetQC>
414    </wsag:QualifyingCondition>
415
416    <wsag:ServiceLevelObjective>
417      <puppetSLO:PuppetSLO>
418        <puppetSLO:Latency>
419          <value>10000</value>
420
421          <puppetSLO:Distribution>
422            <Gaussian>10</Gaussian>
423          </puppetSLO:Distribution>
424        </puppetSLO:Latency>
425      </puppetSLO:PuppetSLO>
426    </wsag:ServiceLevelObjective>
427
428    <wsag:BusinessValueList>
429      <wsag:Penalty>
430        <wsag:AssessmentInterval>
431          <wsag:Count />
432        </wsag:AssessmentInterval>
433
434        <wsag:ValueExpression> 2 </wsag:ValueExpression>
435      </wsag:Penalty>
436    </wsag:BusinessValueList>
437  </wsag:GuaranteeTerm>
438
439  </wsag:All>
440  </wsag:Terms>
441 </wsag:AgreementOffer>
```

5.8 Warehouse

In the following we report the Java source code emulating a Warehouse Web Service. The whole code of the stub was automatically generated by Puppet with the **ambitionMode** enabled. The specification of the warehouse we used in this example is the one given in Chapter 4.6.

```java
1  package services;
2
3  import java.math.BigInteger;
4  import java.util.ArrayList;
```

```java
5    import java.util.Iterator;
6
7    import density.Density;
8
9    public class WarehousePortBindingImpl {
10       /*
11        * The SSM object.
12        */
13       private info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine aMbItIoNssm;
14
15       /*
16        * The simulator is generated
17        */
18       private info.frantzen.testing.ssmsimulator.SSMSimulator aMbItIoNsim;
19
20       private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMessage(
21               info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine ssm,
22               info.frantzen.testing.ssmsimulator.ssm.MessageKind kind,
23               info.frantzen.testing.ssmsimulator.ssm.Operation op)
24               throws Exception {
25           java.util.HashSet<info.frantzen.testing.ssmsimulator.ssm.Message> messages = ssm
26               .getMessages();
27           for (java.util.Iterator it = messages.iterator(); it.hasNext();) {
28               info.frantzen.testing.ssmsimulator.ssm.Message m = (info.frantzen.testing.ssmsimulator.ssm.Message) it
29                   .next();
30               if (m.getKind() != info.frantzen.testing.ssmsimulator.ssm.MessageKind.UNOBSERVABLE) {
31                   if ((m.getKind() == kind) && (m.getOperation().equals(op))) {
32                       return m;
33                   }
34               }
35           }
36           throw new Exception(
37                   "Cannot find the input SSM message belonging to the Operation "
38                       + op.getName() + "!");
39       }
40
41       public WarehousePortBindingImpl() throws Exception {
42           /*
43            * To initialise the Simulator, the following items are needed:
44            */
45
46           /*
47            * The URL of the WSDL file
48            */
49           java.net.URL aMbItIoNWSDLUrl = new java.net.URL("Put here the URL of the Service's WSDL");
50           /*
51            * The name of the WSDL-Service
52            */
53           String aMbItIoNservice = "Put here the name of the Service as in the WSDL";
54           /*
55            * The name of the WSDL-Port
56            */
57           String aMbItIoNport = "Put here the port of the Service as in the WSDL";
58           /*
59            * The URL of the SSM Schema Instance
60            */
61           java.net.URL aMbItIoNSSMUrl = new java.net.URL("Put here the URL of the SSM Schema Instance");
62
63           /*
64            * Now we can generate the SSM object. To do so, we use Zsolt's "Minerva" library
65            */
66           aMbItIoNssm = hu.soft4d.jessi.ssm.SSMHandler.generateSSM(
67                   aMbItIoNWSDLUrl, aMbItIoNSSMUrl, aMbItIoNservice, aMbItIoNport);
68           /*
69            * Before we can use the SSM in the simulator, the parsers have to be attached
70            * to the switches
71            */
72           aMbItIoNssm.attachParsersToSwitches();
73           /*
74            * Next we generate the socket to the treeSolver.
75            */
76           String aMbItIoNsolverHost = "Put here the URL of the Solver";
77           int aMbItIoNsolverPort = "Put here the Port the Solver";
78           java.net.Socket aMbItIoNsolverSocket = new java.net.Socket(
79                   aMbItIoNsolverHost, aMbItIoNsolverPort);
80           /*
81            * The treeSolver sends a welcome message, we remove it from the stream
82            */
83           new java.io.BufferedReader(new java.io.InputStreamReader(
84                   aMbItIoNsolverSocket.getInputStream())).readLine();
85           /*
86            * The simulator can use an external tool to display sequence diagrams
87            * of the messages exchanged. // I skip this here since this takes extra
88            * ressources/
```

30

```
89              */
90
91              /*
92               * The simulator needs a logger to log to
93               */
94              java.util.logging.Logger aMbItIoNlogger = java.util.logging.Logger
95                      .getLogger("");
96
97              /* The simulator is generated */
98
99              aMbItIoNsim = new info.frantzen.testing.ssmsimulator.SSMSimulator(
100                     aMbItIoNssm, aMbItIoNsolverSocket, aMbItIoNlogger);
101
102             /*
103              * If Double variables are used we assume this models money
104              * (experimental). In any case, do this:
105              */
106             info.frantzen.testing.ssmsimulator.types.ST_PseudoPosDouble.postPointLength = 2;
107             /*
108              * Now the Simulator is ready.
109              * -------------------------------------------------------------------------
110              */
111         }
112
113         public services.Quote checkAvail(services.QuoteRequest r)
114                 throws java.rmi.RemoteException {
115             long aMbItIoNinvocationTime = 0;
116             services.Quote aMbItIoNreturnValue;
117             try {
118                 aMbItIoNinvocationTime = System.currentTimeMillis();
119
120                 /*
121                  * Code Generated for Integration with Ambition
122                  */
123                 info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
                        ssmsimulator.ssm.Operation("checkAvail", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
                        REQUESTRESPONSE);
124                 info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage( aMbItIoNssm,
                        info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
125                 info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
                        ssmsimulator.ssm.Valuation();
126                 java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
                        aMbItIoNmessage.getType();
127                 java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
128                 info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
129
130                 /*
131                  * Generated Parameter 0
132                  */
133                 aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
134                 Object[] aMbItIoNparameterValues = new Object[2];
135                 aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(r.
                        getProduct());
136                 aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_PosIntInstance(r.
                        getQuantity());
137                 info.frantzen.testing.ssmsimulator.types.TypeInstance rInstance = new info.frantzen.testing.
                        ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
138                 aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), rInstance);
139
140                 /*
141                  * The valuation is ready, we can construct an instantiated message
142                  */
143                 info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
                        ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);
144
145                 /*
146                  * This instantiated message can now be given to the simulator. Note
147                  * that here the simulator can potentially spot a failure, namely
148                  * when this message is not specified in the SSM! In that sense,
149                  * here we do testing.
150                  */
151                 aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
152
153                 /*
154                  * Ok, the simulator knows the input. Now we need a functionally
155                  * correct response to this call. We first ask the simulator for all
156                  * currently activated output transitions.
157                  */
158                 java.util.ArrayList aMbItIoNoutputs = new java.util.ArrayList(aMbItIoNsim.getCurrentOutputSwitches());
159
160                 /*
161                  * Out of all possible output switches, we randomly choose one and
162                  * check if it has a solution. If yes, we take it. If not, we choose
163                  * randomly the next one.
```

31

```
164                      */
165                     boolean aMbItIoNnoSolutionFound = true;
166                     info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNnextOutput = null;
167                     java.util.Random aMbItIoNrandom = new java.util.Random();
168                     while (!aMbItIoNoutputs.isEmpty() && aMbItIoNnoSolutionFound) {
169                         info.frantzen.testing.ssmsimulator.ssm.Switch aMbItIoNcandidate = (info.frantzen.testing.
                                ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));
170
171                         /*
172                          * try to find a solution, if yes, fine, if not, remove the
173                          * candidate
174                          */
175                         aMbItIoNnextOutput = aMbItIoNsim.findSolution(aMbItIoNcandidate);
176                         if (aMbItIoNnextOutput == null)
177                             aMbItIoNoutputs.remove(aMbItIoNcandidate);
178                         else
179                             aMbItIoNnoSolutionFound = false;
180                     }
181                     if (aMbItIoNnextOutput == null)
182                         throw new Exception("Failure␣in␣SSM!␣No␣output␣for␣synchronous␣input␣specified!");
183                     /*
184                      * Ok, we have now a feasible and functionally correct output:
185                      * nextOutput Before we send this output to the Service out there,
186                      * we tell so to the simulator:
187                      */
188                     aMbItIoNsim.processInstantiatedMessageNoBackup(aMbItIoNnextOutput);
189
190                     /*
191                      * What is left to do, is to map this instantiated message back to a
192                      * real returnValue.
193                      */
194                     info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNreturnMessage = aMbItIoNnextOutput.getMessage()
                                ;
195                     String aMbItIoNreturnVarName = ((info.frantzen.testing.ssmsimulator.ssm.InteractionVariable)
                                aMbItIoNreturnMessage.getType().iterator().next()).getName();
196                     info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
                                getValuation();
197                     info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
                                .getSingleInstance(aMbItIoNreturnVarName);
198                     String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
199                     aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue()
                                ),aMbItIoNarrayRepresentation[1],(Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()),(
                                Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
200                     /*
201                      * Now send the returnValue back to the calling service. That's it.
202                      */
203                 } catch (Exception genericException) {
204                     throw new java.rmi.RemoteException(genericException.getMessage());
205                 }
206                 Density D = new Density();
207                 Double sleepValue = D
208                         .gaussian(25000 - puppet.ambition.Naturals
209                                 .asNatural(aMbItIoNinvocationTime
210                                         - System.currentTimeMillis()));
211                 try {
212                     Thread.sleep(sleepValue.longValue());
213                 } catch (InterruptedException e) {
214                 }
215                 return aMbItIoNreturnValue;
216             }
217
218             public void cancelTransact(int ref) throws java.rmi.RemoteException {
219                 long aMbItIoNinvocationTime = 0;
220                 try {
221                     aMbItIoNinvocationTime = System.currentTimeMillis();
222                     /*
223                      * Code Generated for Integration with Ambition
224                      */
225                     info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
                                ssmsimulator.ssm.Operation(
226                             "cancelTransact",
227                         info.frantzen.testing.ssmsimulator.ssm.OperationKind.ONEWAY);
228                     info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage(
229                             aMbItIoNssm,
230                             info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT,
231                             aMbItIoNoperation);
232                     info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
                                ssmsimulator.ssm.Valuation();
233                     java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
                                aMbItIoNmessage
234                             .getType();
235                     java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
236                     info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
237
```

```
238            /*
239             * Generated Parameter 0
240             */
241            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit
242                  .next();
243            info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
                  ssmsimulator.types.ST_PosIntInstance(
244                  ref);
245            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),
246                  refInstance);
247
248            /*
249             * The valuation is ready, we can construct an instantiated message
250             */
251            info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
                  ssmsimulator.ssm.InstantiatedMessage(
252                  aMbItIoNmessage, aMbItIoNvaluation);
253
254            /*
255             * This instantiated message can now be given to the simulator. Note
256             * that here the simulator can potentially spot a failure, namely
257             * when this message is not specified in the SSM! In that sense,
258             * here we do testing.
259             */
260            aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
261        } catch (Exception genericException) {
262            throw new java.rmi.RemoteException(genericException.getMessage());
263        }
264    }
265
266    static long startWinTimeStamp = System.currentTimeMillis();
267
268    static ArrayList<Long> faultBuffer = new ArrayList<Long>();
269
270    static int executedFault = 0;
271
272    public void orderShipment(int ref, services.Address adr) throws java.rmi.RemoteException {
273        long aMbItIoNinvocationTime = 0;
274        try {
275            aMbItIoNinvocationTime = System.currentTimeMillis();
276            /*
277             * Code Generated for Integration with Ambition
278             */
279            info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
                  ssmsimulator.ssm.Operation("orderShipment", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
                  ONEWAY);
280            info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage( aMbItIoNssm,
                  info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
281            info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
                  ssmsimulator.ssm.Valuation();
282            java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
                  aMbItIoNmessage.getType();
283            java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
284            info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
285
286            /*
287             * Generated Parameter 0
288             */
289            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
290            info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
                  ssmsimulator.types.ST_PosIntInstance(ref);
291            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), refInstance);
292
293            /*
294             * Generated Parameter 1
295             */
296            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
297            Object[] aMbItIoNparameterValues = new Object[2];
298            aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                  getFirstName());
299            aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                  getLastName());
300            info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
                  ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
301            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),adrInstance);
302
303            /*
304             * The valuation is ready, we can construct an instantiated message
305             */
306            info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
                  ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);
307
308            /*
309             * This instantiated message can now be given to the simulator. Note
```

```
310          * that here the simulator can potentially spot a failure, namely
311          * when this message is not specified in the SSM! In that sense,
312          * here we do testing.
313          */
314          aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
315        } catch (Exception genericException) {
316          throw new java.rmi.RemoteException(genericException.getMessage());
317        }
318        long winSize = 120000;
319        int maxFault = 3;
320        long currentTimeStamp = System.currentTimeMillis();
321        for (int i=0; i<faultBuffer.size();i++){
322          if (currentTimeStamp - faultBuffer.get(i) >= winSize){
323            faultBuffer.remove(i);
324          }
325        }
326        if (faultBuffer.size() < maxFault){
327          Density d = new Density();
328          double dv = d.gaussian(100);
329          if (dv > 50) {
330            String fCode = "Server.NoService";
331            String fString = "PUPPET: No target service to invoke!";
332            org.apache.axis.AxisFault fault = new org.apache.axis.AxisFault(
333                fCode, fString, "", null);
334            aMbItIoNsim.undo();
335            faultBuffer.add(currentTimeStamp);
336            throw fault;
337          }
338        }
339      }
340  }
```

References

[1] A. Bertolino, G. D. Angelis, and A. Polini. A QoS Test-bed Generator for Web Services. In *Proc. of the 7th International Conference on Web Engineering 2007 (ICWE 2007)*, volume LNCS series, Como, Italy, 2007. Springer Verlag.

[2] A. Bertolino, G. D. Angelis, and A. Polini. Automatic Generation of Test-beds for Pre-Deployment QoS Evaluation of Web Services. In *Proc. of the 6th International Workshop on Software and Performance (WOSP 2007)*, Buenos Aires, Argentina, 2007. ACM.

[3] A. Bertolino, D. Bianculli, A. Carzaniga, G. De Angelis, I. Forgacs, L. Frantzen, Z. Gere, C. Ghezzi, A. Polini, F. Raimondi, A. Sabetta, and A. Wolf. Test Framework Specification and Architecture. Technical Report Deliverable D4.1, PLASTIC Consortium, March 2007. IST STREP Project.

[4] A. Bertolino, D. Bianculli, A. Carzaniga, G. De Angelis, I. Forgacs, L. Frantzen, Z. Gere, C. Ghezzi, A. Polini, F. Raimondi, A. Sabetta, and A. Wolf. Test Framework Specification and Architecture. Technical Report Deliverable D4.3, PLASTIC Consortium, March 2008. IST STREP Project.

[5] W. Emmerich, F. Raimondi, J. Skene, V. Cortellessa, P. Inverardi, M. Tivoli, D. D. Ruscio, M.Autili, R. Mirandola, V. Grassi, A. Sabetta, J. Gonzales, P. Mazzoleni, and S. Tai. SLA language and analysis techniques for adaptable and resource-aware components. Technical Report Deliverable D2.1, PLASTIC Consortium, March 2007. IST STREP Project.

[6] P. Inverardi, V. Cortellessa, A. Di Marco, M. Autili, et al. Formal description of the PLASTIC conceptual model and of its relationship with the PLASTIC platform toolset. Technical Report Deliverable D1.2, PLASTIC Consortium, March 2008. IST STREP Project.

[7] F. Liotopoulos, S. Tai, J. Sairamesh, H. Eikerling, J. Gonzalez, J. Barra, M. Jazayeri, J. Wuttke, P. Inverardi, V. Cortellessa, A. Di Marco, and M. Autili. Scenarios, Requirements and initial Conceptual Model. Technical Report Deliverable D1.1, PLASTIC Consortium, June 2006. IST STREP Project.

[8] H. Ludwig. WS-Agreement Concepts and Use - Agreement-Based Service-Oriented Architectures. Technical Report RC23949, IBM, May 2006.

[9] J. Skene and W. Emmerich. Engineering runtime requirements: monitoring systems using MDA technologies. 2005.