

PUPPET - User Manual

Guglielmo De Angelis

August 5, 2008

Abstract

This article is the user's guide for PUPPET . Please refer to [4] 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	2
2 Technical info	2
3 Deployment	3
3.1 Install	3
3.2 Configure	4
3.3 Usage	4
4 Tutorial	5
4.1 Terms in the Agreement and Generation Process	5
4.2 The Syntax for the Terms in the WS-Agreement Contracts	5
4.2.1 Qualifying Conditions	5
4.2.2 Service Level Objective	5
4.2.3 Scope	7
4.3 Writing an Agreement	7
4.4 Functional Behavior with Jambition	7
4.5 The Emulation of the Mobility with PUPPET	13
4.5.1 The Configuration of the Mobility Oracle	14
4.6 Example	14
4.6.1 Scenario Description	14
4.6.2 Actors Interactions in the Scenario	17
4.6.3 QoS Properties Definition	17
4.7 Reference To Jambition	18
5 Appendix	18
5.1 Abstract WSDL: WSDoctor	18
5.2 Abstract WSDL: WSCalendar	19
5.3 Abstract WSDL: WSSupervisor	20
5.4 Abstract WSDL: WSPlastic	21
5.5 Abstract WSDL: WSSeguiteService	22
5.6 Abstract WSDL: WSMedicalDevice	24
5.7 WS Agreement	26
5.8 Warehouse	31
References	36

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 [4, 3, 2], 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 provide 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 4.7).

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

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 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 – available at <http://ws.apache.org/axis>.

Apache XMLBeans Libs : xbean.jar – available at <http://xmlbeans.apache.org>.

String Template Libs : stringtemplate.jar – available at <http://www.stringtemplate.org>.

INI4J Libs : ini4j.jar, ini4j-compat.jar – available at <http://ini4j.sourceforge.net>.

Jambition Libs : SSMSimulator.jar, minerva.jar (see <http://plastic.isti.cnr.it/wiki/tools>)

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

Please note that PuppetD4.4.tgz archive does not include these libraries. Launching PUPPET you have to download them separately. Thus, you can either include such libraries in the Java Classpath or copy them in the directory **PuppetD4.4/externalLibs** and use the scripts : puppet.sh (Linux), runPuppet.bat (Windows XP).

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 [4, 3, 2, 6, 7].

Tasks N/A

Bugs This version is affected by the following known bugs. We are keep working on PUPPET in order to completely remove them.

- Using PUPPET , please do not refer to any path that includes any blank. For example, you should not use the path : “C:\Documents and Settings*”.
- Occasionally, the generation process may end with the message “parsing done” even though the executions failed. This is due to some exceptions that was not correctly caught.

Patches N/A

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

3 Deployment

3.1 Install

Unzip the archive **PuppetD4.4.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 **PuppetD4.4/doc** contains this manual. The directory **PuppetD4.4/xml** contains the definitions mapping of the WS-Agreement statements into the Java code. The directory **PuppetD4.4/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 the external library to the Java CLASSPATH variable. **PuppetD4.3/puppet.sh** and **PuppetD4.3/runPuppet.bat** are executable batch scripts that set the Java CLASSPATH variable and run PUPPET on a given input file. When you use such scripts, please copy the external libraries into the directory **PuppetD4.4/externalLibs**.

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 storing 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.

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

trueTermsFilename : It is the name of the file storing 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 storing the WS-Agreement file. Required.

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

qcMappingFilename : It is the path to the file storing the template mapping of the Qualifying Conditions 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.

sloMappingFilename : It is the path to the file storing 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.

mobilityMode : If it is set to “**on**” enables the emulation of the mobility of the nodes that are supposed to deploy the services emulated by means of the stubs. By default this flag 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**¹ on a given <IniConfigurationFile>.

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

Table 1: Enabling the code generation in PUPPET

4 Tutorial

4.1 Terms in the Agreement and Generation Process

In WS-Agreement [11], 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.

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 [11]. 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 [11].

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.

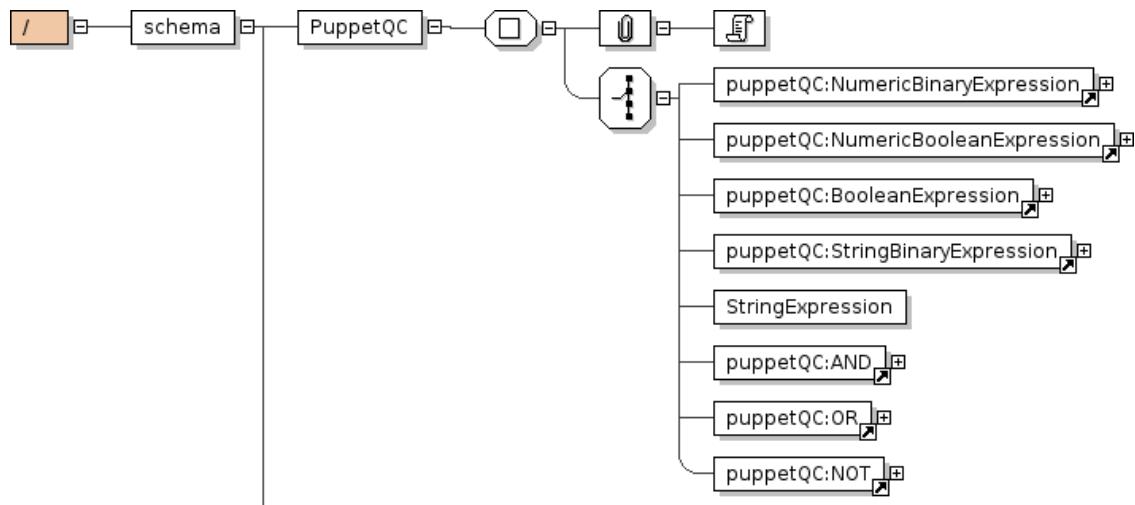


Figure 1: Expressions in the Qualifying Condition

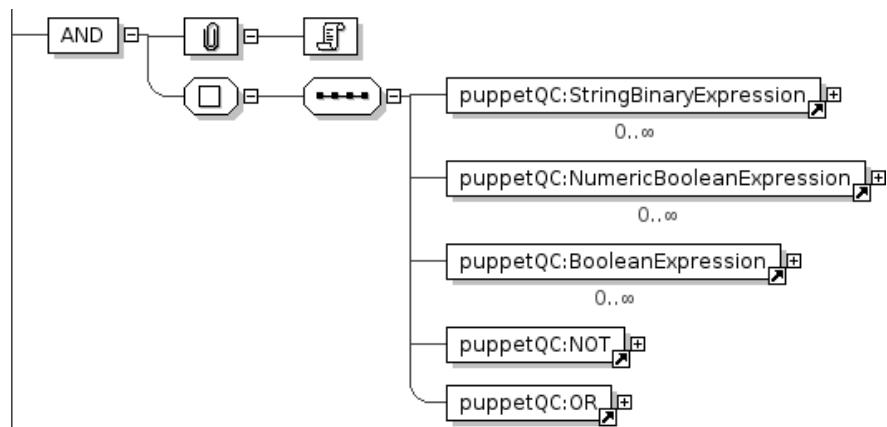


Figure 2: AND in the Qualifying Condition

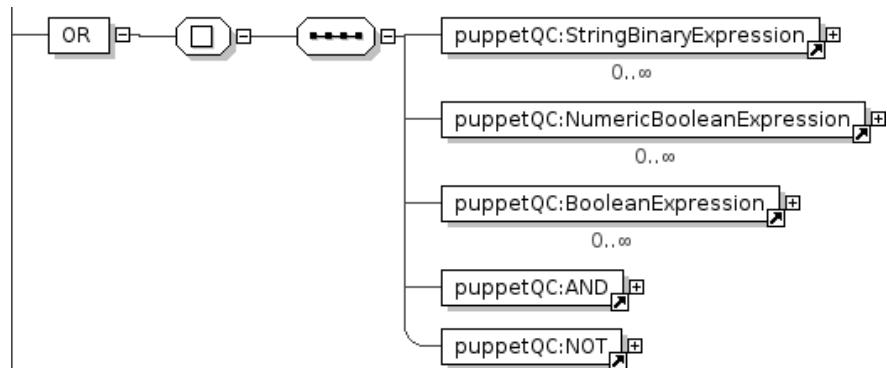


Figure 3: OR in the Qualifying Condition

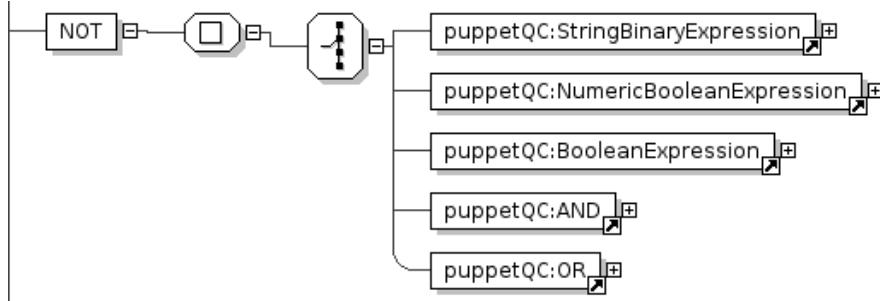


Figure 4: NOT in the Qualifying Condition

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 [11], 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 [4]. The latter is to exploit the PLASTIC's editor of SLA as explained in the following.

The PLASTIC conceptual model [10] 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 [10] and then refined in D1.2 [9].

According to the conceptual model, [8] defines with SLang [12] 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 [8] 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 4.7) 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 4.7, 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).

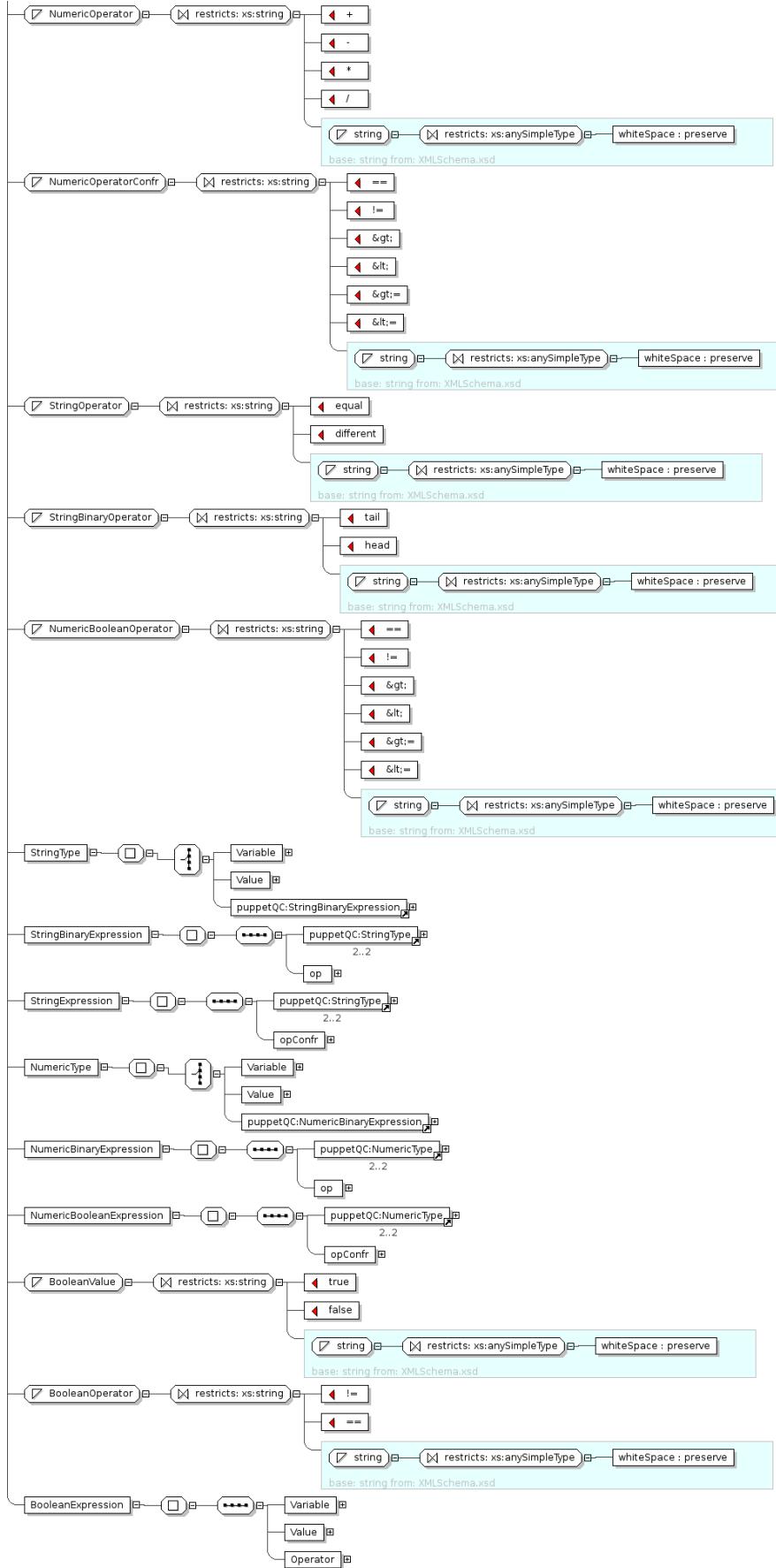


Figure 5: Operators in the Qualifying Condition

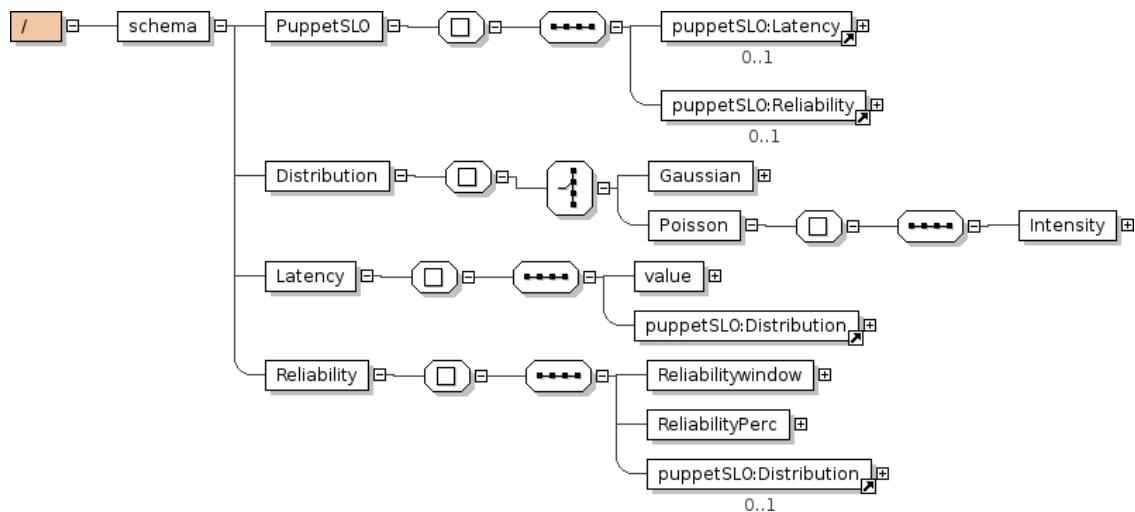


Figure 6: Extra-Functional Properties in the Service Level Objective

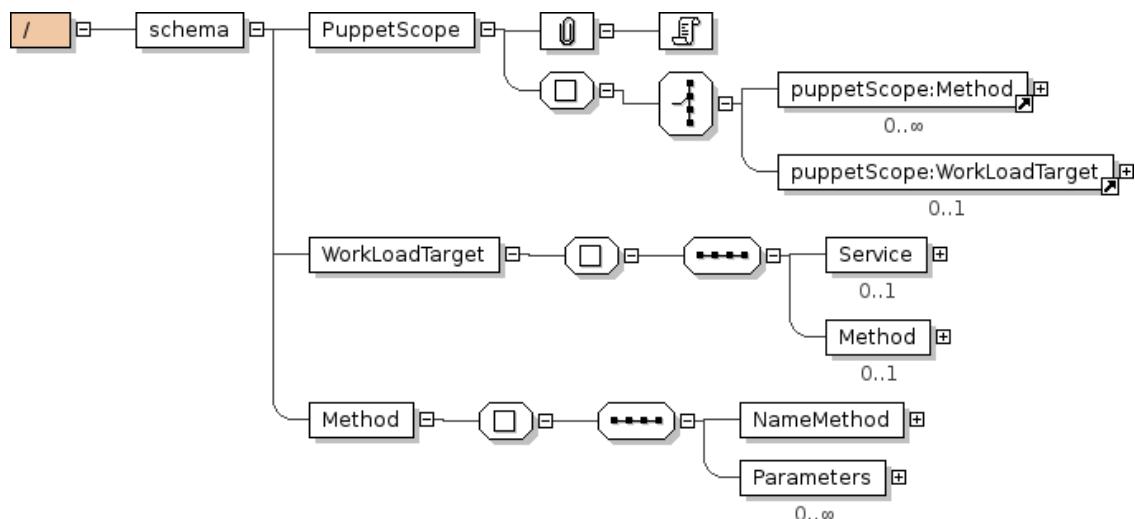


Figure 7: Defining the Scope of a Term

```

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 aMbItIoNsims;
19
20 private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMesssage(
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  * Now we can generate the SSM object. To do so, we use Zsolt's "Minerva" library
64  */
65 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      * The simulator needs a logger to log to
92      */
93     java.util.logging.Logger aMbItIoNlogger = java.util.logging.Logger
94         .getLogger("");
95
96     /* The simulator is generated */
97
98     aMbItIoNsim = new info.frantzen.testing.ssmsimulator.SSMSimulator(
99         aMbItIoNssm, aMbItIoNsolverSocket, aMbItIoNlogger);
100
101    /*
102     * If Double variables are used we assume this models money
103     * (experimental). In any case, do this:
104     */
105     info.frantzen.testing.ssmsimulator.types.ST_PseudoPosDouble.postPointLength = 2;
106     /*
107     * Now the Simulator is ready.
108     * -----
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.
280             ssmsimulator.ssm.Operation("orderShipment", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
281             ONEWAY);
282         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessages( aMbItIoNssm,
283             info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
284         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
285             ssmsimulator.ssm.Valuation();
286         java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
287             aMbItIoNmessage.getType();
288         java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
289         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.
291         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) aMbItIoNt.next();
297     Object[] aMbItIoNparameterValues = new Object[2];
298     aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
299         getFirstName());
300     aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
301         getLastName());
300     info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
301         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.
307         ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmmessage, aMbItIoNvaluation);
308
309     /*
310      * This instantiated message can now be given to the simulator. Note
311      * that here the simulator can potentially spot a failure, namely
312      * when this message is not specified in the SSM! In that sense,
313      * here we do testing.
314     */
314     aMbItIoNsims.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(aMbItIoNsims.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.
170             ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));
171
172         /*
173          * try to find a solution, if yes, fine, if not, remove the
174          * candidate
175        */
175         aMbItIoNnextOutput = aMbItIoNsims.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     ;
196     String aMbItIoNreturnVarName = ((info.frantzen.testing.ssmsimulator.ssm.InteractionVariable)
197         aMbItIoNreturnMessage.getType().iterator().next()).getName();
198     info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
199     getValuation();
200     info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
201     .getSingleInstance(aMbItIoNreturnVarName);
202     String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
203     aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue(),
204     ), aMbItIoNarrayRepresentation[1], (Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()), (
205     Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
206     /*
207     * Now send the returnValue back to the calling service. That's it.
208     */

```

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 4.7). The whole code of the stub was automatically generated by Puppet with the **ambitionMode** enabled.

4.5 The Emulation of the Mobility with PUPPET

PUPPET supports the validation of service-oriented applications, aiming at evaluating the desired QoS characteristics for a specific service under development before it is deployed.

In [7] PUPPET was equipped with the capability to simulate the runtime movement of the nodes that host the generated stubs. Indeed, the QoS properties of each testing environment that PUPPET generates could depend on the mobility models of the devices where the services of the testbed are deployed.

The benefit of introducing a mobility model in PUPPET is twofold. On the one hand, the proposed approach is able to validate the QoS features of the Service Under Test (SUT) by taking into account the availability of the other interacting services. In this sense, PUPPET assumes that a service is available in a network scenario if, according to the used mobility model, the node where this service is deployed is reachable. On the other hand, PUPPET permits to use in the testing phase the same mobility model that can be used during the design and analysis phases.

In this version, PUPPET uses NS-2 [1] in order to generate the mobility model for the network nodes. The movement of the nodes are defined by means of NS-2's `setdest` tool. The results of an NS-2 simulations with the movement specifications can be shown in a tabular form by means of trace files. In a trace file, a timestamped line is produced for each data packet that travels on the network. The trace file reports the location of each node at a given time interval (for example each second). Thus, specifying for each node the initial position, and its speed, it is possible to trace the movement of the node during the whole simulation.

PUPPET emulates the mobility of the remote services equipping the generated stub with a hook able to interact with a mobility oracle. A mobility oracle is a service that takes in input a mobility model produced by the NS-2 simulator. This service exports an operation that taking in input the name of the nodes where two services are deployed, returns an index in [0..1] measuring the damping factor in the QoS levels due to their relative distance. The index is 0 if the nodes are in the same position, else it is 1 if the two nodes (and the services deployed on them) are not visible each other. Values in (0..1) give indications on their relative distance.

PUPPET injects into the generated stub a portion of Java code that interprets the damp index received from the mobility oracle. If the returned damp index is 1, a failure is generated as an exception raised by the platform hosting the Web Service stub. In this manner, the reliability of the SUT will depend not only on the reliability exposed in the QoS agreements by the composed services, but also on the availability of the

services it composes, as simulated by the mobility model of a given scenario.

When the damp index returned by the mobility oracle is included in (0..1), the Java code injected into the stub behaves emulating an additional latency that is function of the damp index. In this manner, the interactions between services deployed on different nodes that can move in a space, will affect the response time. The delta introduced in the latency is function of the damp index.

Finally, the mobility oracle returns a 0 damp index when two mobile devices are in the same location. In this case, the extra functional behavior defined in the agreement is not modified.

Note that both the functions computing the metric for the visibility and the damping factor can be specifically defined for each scenario. PUPPET provides an abstract specification of the mobility oracle that can be extended instantiating both the metric for the mobility, and the damping factor according to the requirements of each scenario.

PUPPET includes a default implementation for the mobility oracle where the visibility function is expressed in terms of euclidean distance between the two nodes. Two nodes will be visible if and only if their relative distance is below a given threshold. Also, the damping index influencing the QoS agreed by the services will increase linear to the euclidean distance of the nodes.

For the sake of completeness, we note that in this version of PUPPET one only instance of the mobility oracle is shared among all the stubs of a testing scenario.

4.5.1 The Configuration of the Mobility Oracle

The Mobility Oracle (implemented in `puppet.jar` as `AbstractTraceReader` and `TraceReader`) can be configured by means of a INI Configuration file. In such file, the mobility oracle looks for a section named **[mobilityConfiguration]**. The parameters that could be specified into the input configuration file are:

nodes : It is the number of nodes which movement is listed in a NS-2 trace file, and it is emulated by the mobility oracle. By default it is set to "1". Optional.

maxRadius : Given a metric defining the *visibility* among two nodes, this index define the bound maximum for such metric. By default it is set to "400d"². Optional.

traceFilename : It is the path to the NS-2 trace file that describes the mobility of the nodes in the scenario. By default it is set to "./**mobilityConfiguration.ini**". Optional.

4.6 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 [8] for details).

4.6.1 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 features that it exports.

Specifically the operation **notifyAlarm** is aimed at both collecting and processing 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 has 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)

WSSeguiteService : This Web Service represents the current eHealth application. It exports the following operations: **notifyAlarm**, **getResidentialGatewayIP**, **getContactDecivelP**. **notifyAlarm**, takes as input the name of the Residential Gateway where the alarm comes from, the kind of the alarm and the date

²Note that the visibility metric in `TraceReader` is implemented as euclidean distance

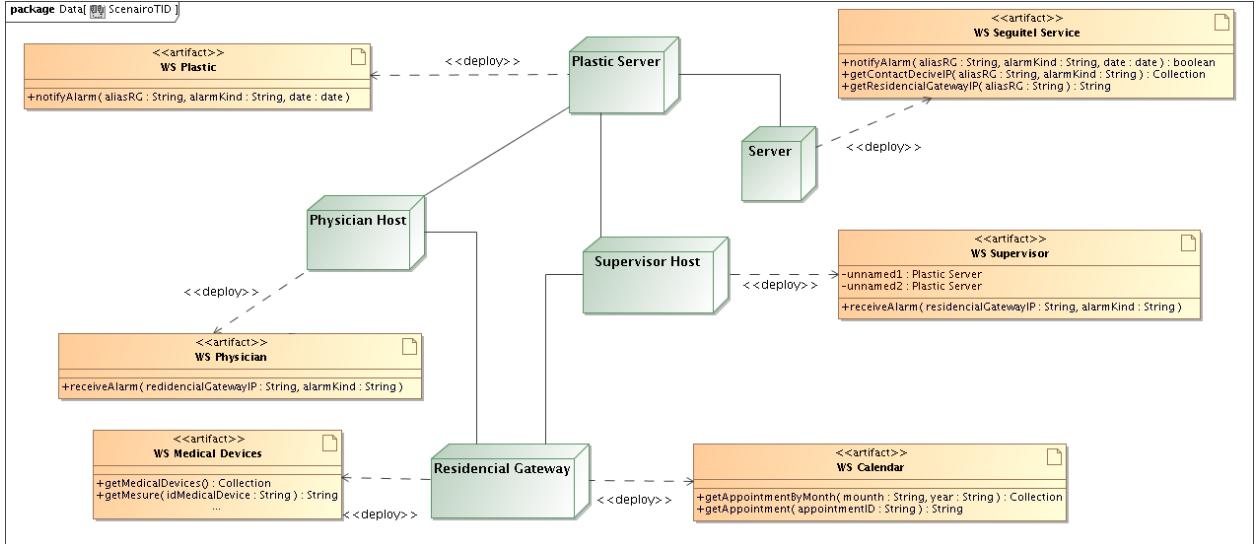


Figure 8: Scenario 3 Deployment Diagram

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.

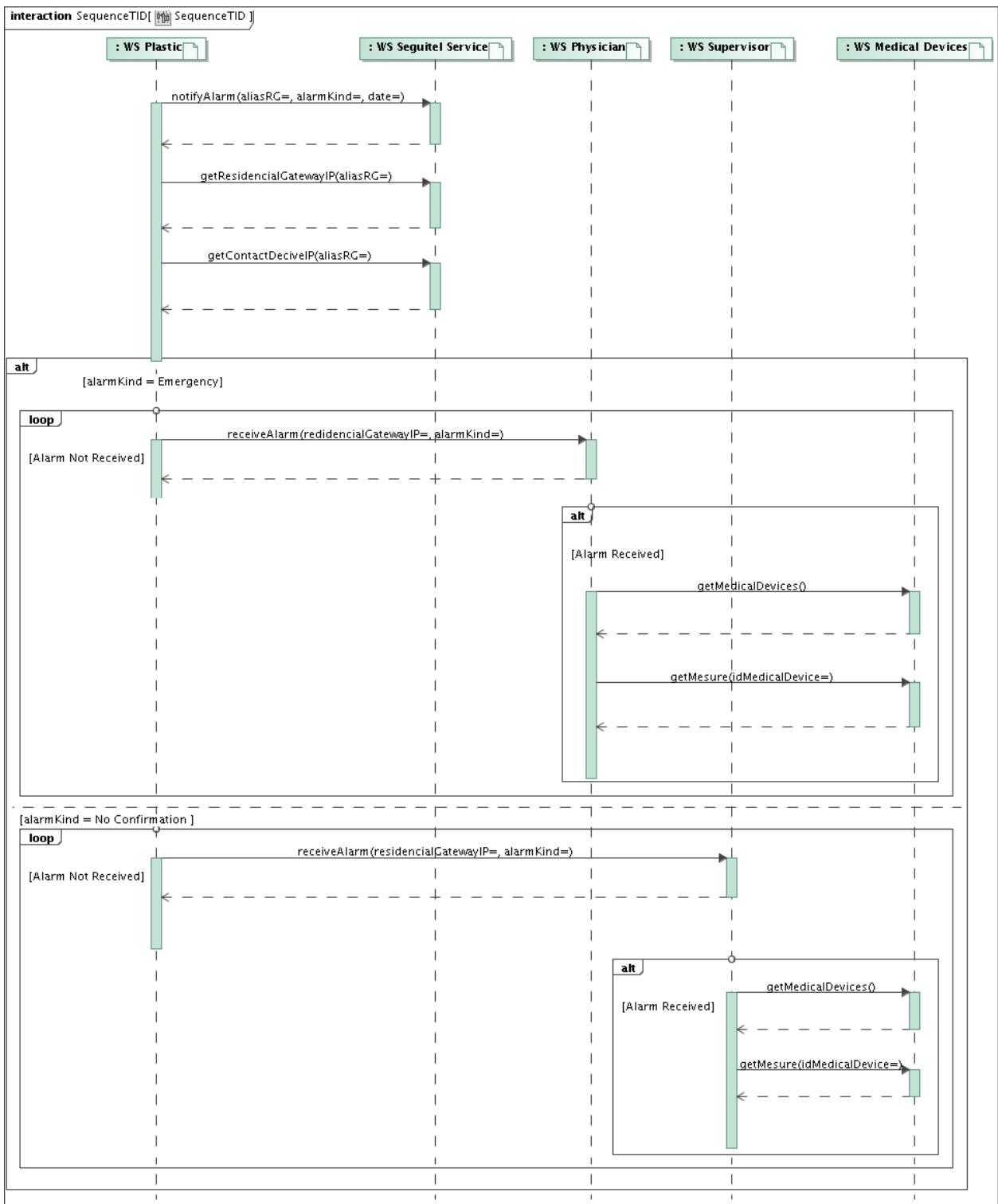


Figure 9: Scenario 3 Sequence Diagram

4.6.2 Actors Interactions in the Scenario

This section provides a brief description on how the Web Services described in Section 4.6.1 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.

4.6.3 QoS Properties Definition

As described in Section 4.6.1, 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.

	Latency (msec)	Reliability		Conditions
WSSequitelService:getContactDeciveIP	2000			alarmKind="Emergency"
WSSequitelService: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

Is it clear that the QoS properties of the service could depend on where the service is 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.

The QoS levels admitted in the scenario here considered are formalized in an agreement. Table 2³ 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.

4.7 Reference To Jambition

For any reference to Jambition in this document, please refer to [5].

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.
3    apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSDoctor" xmlns:intf="http://
4      localhost:8080/axis/services/WSDoctor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl=
5        "http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http:
6          //www.w3.org/2001/XMLSchema">
7  <!--WSDL created by Apache Axis version: 1.4
8  Built on Apr 22, 2006 (06:55:48 PDT)-->
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:message name="receiveAlarmResponse">
19
20 </wsdl:message>
21
22 <wsdl:portType name="WSDoctor">
23
24   <wsdl:operation name="receiveAlarm" parameterOrder="residencialGatewayIP,alarmGender">
25
26     <wsdl:input message="impl:receiveAlarmRequest" name="receiveAlarmRequest"/>
27
28     <wsdl:output message="impl:receiveAlarmResponse" name="receiveAlarmResponse"/>
29
30   </wsdl:operation>
31
32 </wsdl:portType>
33
34 <wsdl:binding name="WSDoctorSoapBinding" type="impl:WSDoctor">
35
36   <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
37
38   <wsdl:operation name="receiveAlarm">
39
40     <wsdlsoap:operation soapAction="" />
41
42     <wsdl:input name="receiveAlarmRequest">
43
44       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
45         use="encoded"/>
46
47     </wsdl:input>
48
49     <wsdl:output name="receiveAlarmResponse">
50
51       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
52         localhost:8080/axis/services/WSDoctor" use="encoded"/>
53
54     </wsdl:output>
55
56   </wsdl:operation>
57
58 </wsdl:binding>
59
60 <wsdl:service name="WSDoctorService">
```

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

```

55 <wsdl:port binding="impl:WSDoctorSoapBinding" name="WSDoctor">
56   <wsdlsoap:address location="http://localhost:8080/axis/services/WSDoctor"/>
57 </wsdl:port>
58 </wsdl:service>
59 </wsdl:definitions>
60
61
62
63
64

```

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-
3    .apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSCalendar" xmlns:intf="http://
4      localhost:8080/axis/services/WSCalendar" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
5    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/">
6    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
7    <!--WSDL created by Apache Axis version: 1.4
8    Built on Apr 22, 2006 (06:55:48 PDT)-->
9    <wsdl:types>
10   <schema targetNamespace="http://localhost:8080/axis/services/WSCalendar" xmlns="http://www.w3.org/2001/
11     XMLSchema">
12     <import namespace="http://xml.apache.org/xml-soap"/>
13     <import namespace="http://schemas.xmlsoap.org/soap/encoding"/>
14     <complexType name="ArrayOf_xsd_anyType">
15       <complexContent>
16         <restriction base="soapenc:Array">
17           <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
18         </restriction>
19       </complexContent>
20     </complexType>
21   </schema>
22   <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
23     <import namespace="http://localhost:8080/axis/services/WSCalendar"/>
24     <import namespace="http://schemas.xmlsoap.org/soap/encoding"/>
25     <complexType name="Vector">
26       <sequence>
27         <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
28       </sequence>
29     </complexType>
30   </schema>
31 </wsdl:types>
32
33 <wsdl:message name="getAppointmentByMonthRequest">
34   <wsdl:part name="month" type="soapenc:string"/>
35   <wsdl:part name="year" type="soapenc:string"/>
36 </wsdl:message>
37
38 <wsdl:message name="getAppointmentRequest ">
39   <wsdl:part name="apointmentID" type="soapenc:string"/>
40 </wsdl:message>
41
42 <wsdl:message name="getAppointmentByMonthResponse">
43   <wsdl:part name="getAppointmentByMonthReturn" type="impl:ArrayOf_xsd_anyType"/>
44 </wsdl:message>
45
46 <wsdl:message name="getAppointmentResponse">
47   <wsdl:part name="getAppointmentReturn" type="soapenc:string"/>
48 </wsdl:message>
49
50 <wsdl:portType name="WSCalendar">
51
52   <wsdl:operation name="getAppointmentByMonth" parameterOrder="month,year">
53     <wsdl:input message="impl:getAppointmentByMonthRequest" name="getAppointmentByMonthRequest"/>
54     <wsdl:output message="impl:getAppointmentByMonthResponse" name="getAppointmentByMonthResponse"/>
55   </wsdl:operation>
56
57   <wsdl:operation name="getAppointment" parameterOrder="apointmentID">
58     <wsdl:input message="impl:getAppointmentRequest" name="getAppointmentRequest"/>
59     <wsdl:output message="impl:getAppointmentResponse" name="getAppointmentResponse"/>
60   </wsdl:operation>
61
62
63
64
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
83       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
84         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://
90       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"
102      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://
108      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

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSSupervisor" xmlns:apachesoap="http://
2    xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSupervisor" xmlns:intf="http://
2      localhost:8080/axis/services/WSSupervisor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
2      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
2      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"
41         use="encoded" />
42
43   </wsdl:input>
44
45   <wsdl:output name="receiveAlarmResponse">
46
47     <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
48       localhost:8080/axis/services/WSSupervisor" use="encoded" />
49
50   </wsdl:output>
51
52 </wsdl:operation>
53
54 </wsdl:binding>
55
56 <wsdl:service name="WSSupervisorService">
57
58   <wsdl:port binding="impl:WSSupervisorSoapBinding" name="WSSupervisor">
59
60     <wsdlsoap:address location="http://localhost:8080/axis/services/WSSupervisor" />
61
62   </wsdl:port>
63
64 </wsdl:service>
65
66 </wsdl:definitions>

```

5.4 Abstract WSDL: WSPlastic

```
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   <wsdl:input message="impl:notifyAlarmRequest" name="notifyAlarmRequest"/>
24   <wsdl:output message="impl:notifyAlarmResponse" name="notifyAlarmResponse"/>
25 </wsdl:operation>
26 </wsdl:portType>
27 </wsdl:binding name="WSPlasticSoapBinding" type="impl:WSPlastic">
28   <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
29   <wsdl:operation name="notifyAlarm">
30     <wsdlsoap:operation soapAction="" />
31     <wsdl:input name="notifyAlarmRequest">
32       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
33         use="encoded"/>
34     </wsdl:input>
35     <wsdl:output name="notifyAlarmResponse">
36       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
37         localhost:8080/axis/services/WSPlastic" use="encoded"/>
38     </wsdl:output>
39   </wsdl:operation>
40 </wsdl:binding>
41 <wsdl:service name="WSPlasticService">
42   <wsdl:port binding="impl:WSPlasticSoapBinding" name="WSPlastic">
43     <wsdlsoap:address location="http://localhost:8080/axis/services/WSPlastic"/>
44   </wsdl:port>
45 </wsdl:service>
46 </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="
3    http://xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSEguitelService"
4    xmlns:intf="http://localhost:8080/axis/services/WSSEguitelService" xmlns:soapenc="http://schemas.xmlsoap.
5    org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.
6    org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
7  <!--WSDL created by Apache Axis version: 1.4
8  Built on Apr 22, 2006 (06:55:48 PDT)-->
9  <wsdl:types>
10   <schema targetNamespace="http://localhost:8080/axis/services/WSSEguitelService" xmlns="http://www.w3.org/2001/
11     XMLSchema">
12     <import namespace="http://xml.apache.org/xml-soap"/>
13     <import namespace="http://schemas.xmlsoap.org/soap/encoding"/>
14     <complexType name="ArrayOf_xsd_anyType">
15       <complexContent>
16         <restriction base="soapenc:Array">
17           <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
18         </restriction>
19       </complexContent>
20     </complexType>
21   </schema>
22   <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
23     <import namespace="http://localhost:8080/axis/services/WSSEguitelService"/>
24     <import namespace="http://schemas.xmlsoap.org/soap/encoding"/>
25     <complexType name="Vector">
26       <sequence>
27         <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
28       </sequence>
29     </complexType>
30   </schema>
31 </wsdl:types>
32 <wsdl:message name="getResidencialGatewayIPResponse">

```

```

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

- 31 </wsdl:message>
- 32 <wsdl:message name="getResidencialGatewayIPRequest">
- 33 <wsdl:part name="aliasRG" type="soapenc:string"/>
- 34 </wsdl:message>
- 35 <wsdl:message name="notifyAlarmRequest">
- 36 <wsdl:part name="aliasRG" type="soapenc:string"/>
- 37 <wsdl:part name="alarmGender" type="soapenc:string"/>
- 38 <wsdl:part name="date" type="xsd:dateTime"/>
- 39 </wsdl:message>
- 40 <wsdl:message name="getConnectedDeviceIPRequest">
- 41 <wsdl:part name="aliasRG" type="soapenc:string"/>
- 42 <wsdl:part name="alarmGender" type="soapenc:string"/>
- 43 </wsdl:message>
- 44 <wsdl:message name="notifyAlarmResponse">
- 45 <wsdl:part name="notifyAlarmReturn" type="xsd:boolean"/>
- 46 </wsdl:message>
- 47 <wsdl:message name="getConnectedDeviceIPResponse">
- 48 <wsdl:part name="getConnectedDeviceIPReturn" type="impl:ArrayOf_xsd_anyType"/>
- 49 </wsdl:message>
- 50 <wsdl:portType name="WSSeguitelService">
- 51 <wsdl:operation name="notifyAlarm" parameterOrder="aliasRG_alarmGender_date">
- 52 <wsdl:input message="impl:notifyAlarmRequest" name="notifyAlarmRequest"/>
- 53 <wsdl:output message="impl:notifyAlarmResponse" name="notifyAlarmResponse"/>
- 54 </wsdl:operation>
- 55 <wsdl:operation name="getConnectedDeviceIP" parameterOrder="aliasRG_alarmGender">
- 56 <wsdl:input message="impl:getConnectedDeviceIPRequest" name="getConnectedDeviceIPRequest"/>
- 57 <wsdl:output message="impl:getConnectedDeviceIPResponse" name="getConnectedDeviceIPResponse"/>
- 58 </wsdl:operation>
- 59 <wsdl:operation name="getResidencialGatewayIP" parameterOrder="aliasRG">
- 60 <wsdl:input message="impl:getResidencialGatewayIPRequest" name="getResidencialGatewayIPRequest"/>
- 61 <wsdl:output message="impl:getResidencialGatewayIPResponse" name="getResidencialGatewayIPResponse"/>
- 62 </wsdl:operation>
- 63 </wsdl:portType>
- 64 <wsdl:binding name="WSSeguitelServiceSoapBinding" type="impl:WSSeguitelService">
- 65 <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
- 66 <wsdl:operation name="notifyAlarm">
- 67 <wsdlsoap:operation soapAction="" />
- 68 <wsdl:input name="notifyAlarmRequest">
- 69 <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario" use="encoded"/>
- 70 </wsdl:input>

```
112 <wsdl:output name="notifyAlarmResponse">
113
114     <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
115         localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
116
117 </wsdl:output>
118
119 </wsdl:operation>
120
121 <wsdl:operation name="getConnectedDeviceIP">
122
123     <wsdlsoap:operation soapAction="" />
124
125     <wsdl:input name="getConnectedDeviceIPRequest">
126
127         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
128             use="encoded"/>
129
130     </wsdl:input>
131
132     <wsdl:output name="getConnectedDeviceIPResponse">
133
134         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
135             localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
136
137     </wsdl:output>
138
139 </wsdl:operation>
140
141 <wsdl:operation name="getResidencialGatewayIP">
142
143     <wsdlsoap:operation soapAction="" />
144
145     <wsdl:input name="getResidencialGatewayIPRequest">
146
147         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
148             use="encoded"/>
149
150     </wsdl:input>
151
152     <wsdl:output name="getResidencialGatewayIPResponse">
153
154         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
155             localhost:8080/axis/services/WSSeguitelService" use="encoded"/>
156
157     </wsdl:output>
158
159 </wsdl:operation>
160
161 </wsdl:binding>
162
163 <wsdl:service name="WSSeguitelServiceService">
164
165     <wsdl:port binding="impl:WSSeguitelServiceSoapBinding" name="WSSeguitelService">
166
167         <wsdlsoap:address location="http://localhost:8080/axis/services/WSSeguitelService"/>
168
169     </wsdl:port>
170
171 </wsdl:service>
172
173 </wsdl:definitions>
```

5.6 Abstract WSDL: WSMedicalDevice

```
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"
81         use="encoded" />
82
83     </wsdl:input>
84
85     <wsdl:output name="getMedicalDevicesResponse">
86
87       <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
88         localhost:8080/axis/services/WSMedicalDevice" use="encoded" />
89
90     </wsdl:output>
91
92   </wsdl:operation>
93
94   <wsdl:operation name="getMeasure">
95     <wsdlsoap:operation soapAction="" />

```

```

96     <wsdl:input name="getMeasureRequest">
97
98         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
99             use="encoded"/>
100
101     </wsdl:input>
102
103     <wsdl:output name="getMeasureResponse">
104
105         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
106             localhost:8080/axis/services/WSMedicalDevice" use="encoded"/>
107
108     </wsdl:output>
109
110 </wsdl:operation>
111
112 </wsdl:binding>
113
114 <wsdl:service name="WSMedicalDeviceService">
115
116     <wsdl:port binding="impl:WSMedicalDeviceSoapBinding" name="WSMedicalDevice">
117
118         <wsdlsoap:address location="http://localhost:8080/axis/services/WSMedicalDevice"/>
119
120     </wsdl:port>
121
122 </wsdl:service>
123
124 </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:puppetsSLO="http://setest0.isti.cnr.it/puppetsSLO"
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
34                         <op>equal</op>
35
36                         <puppetQC:StringType>
37                             <Value>Emergency</Value>
38                         </puppetQC:StringType>
39                     </puppetQC:StringBinaryExpression>
40                 </puppetQC:PuppetQC>
41             </wsag:QualifyingCondition>
42
43             <wsag:ServiceLevelObjective>
44                 <puppetsSLO:PuppetsSLO>
45                     <puppetsSLO:Latency>
46                         <value>2000</value>
47
48                         <puppetSLO:Distribution>
49                             <Gaussian>10</Gaussian>
50                         </puppetSLO:Distribution>
51                     </puppetSLO:Latency>
```

```

52      </puppetSLO:PuppetsSLO>
53  </wsag:ServiceLevelObjective>
54
55  <wsag:BusinessValueList>
56    <wsag:Penalty>
57      <wsag:AssessmentInterval>
58        <wsag:Count />
59      </wsag:AssessmentInterval>
60
61      <wsag:ValueExpression> 2 </wsag:ValueExpression>
62    </wsag:Penalty>
63  </wsag:BusinessValueList>
64  <wsag:GuaranteeTerm>
65    <wsag:GuaranteeTerm wsag:Name="ContactDeciveIP-Term2"
66      wsag:Obligated="ServiceProvider">
67      <wsag:ServiceScope wsag:ServiceName="WSSeguitelService">
68        <puppetScope:PuppetScope>
69          <puppetScope:Method>
70            <NameMethod>getContactDeciveIP</NameMethod> -->
71            <NameMethod>getConnectedDeviceIP</NameMethod>
72          </puppetScope:Method>
73        </puppetScope:PuppetScope>
74      </wsag:ServiceScope>
75
76  <wsag:QualifyingCondition>
77    <puppetQC:PuppetQC>
78      <puppetQC:StringBinaryExpression>
79        <puppetQC:StringType>
80          <Variable>alarmGender</Variable>
81        </puppetQC:StringType>
82
83        <op>equal</op>
84
85        <puppetQC:StringType>
86          <Value>No Confirmation</Value>
87        </puppetQC:StringType>
88      </puppetQC:StringBinaryExpression>
89    </puppetQC:PuppetQC>
90  </wsag:QualifyingCondition>
91
92  <wsag:ServiceLevelObjective>
93    <puppetSLO:PuppetSLO>
94      <puppetSLO:Latency>
95        <value>1000</value>
96
97        <puppetSLO:Distribution>
98          <Gaussian>10</Gaussian>
99        </puppetSLO:Distribution>
100      </puppetSLO:Latency>
101    </puppetSLO:PuppetSLO>
102  </wsag:ServiceLevelObjective>
103
104  <wsag:BusinessValueList>
105    <wsag:Penalty>
106      <wsag:AssessmentInterval>
107        <wsag:Count />
108      </wsag:AssessmentInterval>
109
110      <wsag:ValueExpression> 2 </wsag:ValueExpression>
111    </wsag:Penalty>
112  </wsag:BusinessValueList>
113  <wsag:GuaranteeTerm>
114    <wsag:ExactlyOne>
115      <wsag:GuaranteeTerm wsag:Name="AlarmDoctor-Term1"
116        wsag:Obligated="ServiceProvider">
117        <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      <wsag:ValueExpression> 2 </wsag:ValueExpression>
221      </wsag:Penalty>
222      </wsag:BusinessValueList>
223  </wsag:GuaranteeTerm>
224  </wsag:ExactlyOne>
225  <wsag:ExactlyOne>
226    <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term1"
227      wsag:Obligated="ServiceProvider">
228      <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
229        <puppetScope:PuppetScope>
230          <puppetScope:Method>
231            <NameMethod>receiveAlarm</NameMethod>
232          </puppetScope:Method>
233        </puppetScope:PuppetScope>
234      </wsag:ServiceScope>
235
236    <wsag:QualifyingCondition>
237      <puppetQC:PuppetQC>
238        <puppetQC:StringBinaryExpression>
239          <puppetQC:StringType>
240            <Variable>deployedOn</Variable>
241          </puppetQC:StringType>
242
243        <op>equal</op>
244
245        <puppetQC:StringType>
246          <Value>MobileNode</Value>
247        </puppetQC:StringType>
248        </puppetQC:StringBinaryExpression>
249      </puppetQC:PuppetQC>
250    </wsag:QualifyingCondition>
251
252    <wsag:ServiceLevelObjective>
253      <puppetSLO:PuppetSLO>
254        <puppetSLO:Latency>
255          <value>10000</value>
256
257          <puppetSLO:Distribution>
258            <Gaussian>10</Gaussian>
259          </puppetSLO:Distribution>
260        </puppetSLO:Latency>
261        <puppetSLO:Reliability>
262          <Reliabilitywindow>30000</Reliabilitywindow>
263
264          <ReliabilityPerc>5</ReliabilityPerc>
265
266          <puppetSLO:Distribution>
267            <Gaussian>100</Gaussian>
268          </puppetSLO:Distribution>
269        </puppetSLO:Reliability>
270      </puppetSLO:PuppetSLO>
271    </wsag:ServiceLevelObjective>
272
273    <wsag:BusinessValueList>
274      <wsag:Penalty>
275        <wsag:AssessmentInterval>
276          <wsag:Count />
277        </wsag:AssessmentInterval>
278
279        <wsag:ValueExpression> 2 </wsag:ValueExpression>
280        </wsag:Penalty>
281      </wsag:BusinessValueList>
282    </wsag:GuaranteeTerm>
283  <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term2"
284    wsag:Obligated="ServiceProvider">
285    <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
286      <puppetScope:PuppetScope>
287        <puppetScope:Method>
288          <NameMethod>receiveAlarm</NameMethod>
289        </puppetScope:Method>
290      </puppetScope:PuppetScope>
291    </wsag:ServiceScope>
292
293    <wsag:QualifyingCondition>
294      <puppetQC:PuppetQC>
295        <puppetQC:StringBinaryExpression>
296          <puppetQC:StringType>
297            <Variable>deployedOn</Variable>
298          </puppetQC:StringType>
299
300        <op>equal</op>
301
302        <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 4.7.

```

1 package services;
2
3 import java.math.BigInteger;
4 import java.util.ArrayList;
5 import java.util.Iterator;
6
7 import density.Density;
8
9 public class WarehousePortBindingImpl {
10
11   /**
12    * The SSM object.
13    */
14   private info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine aMbItIoNssm;
15
16   /**
17    * The simulator is generated
18    */
19   private info.frantzen.testing.ssmsimulator.SSMSimulator aMbItIoNs;
20
21   private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMesssage(
22     info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine ssm,
23     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/
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.
124         ssmsimulator.ssm.Operation("checkAvail", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
125             REQUESTRESPONSE);
126     info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessages( aMbItIoNssm,
127         info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
128     info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
129         ssmsimulator.ssm.Valuation();
130     java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
131         aMbItIoNmessage.getType();
132     java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
133     info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
134
135     /*
136      * Generated Parameter 0
137      */
138     aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
139     Object[] aMbItIoNparameterValues = new Object[2];
140     aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(r.
141         getProduct());
142     aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_PosIntInstance(r.
143         getQuantity());
144     info.frantzen.testing.ssmsimulator.types.TypeInstance rInstance = new info.frantzen.testing.
145         ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
146     aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), rInstance);
147
148     /*
149      * The valuation is ready, we can construct an instantiated message
150      */
151     info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
152         ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);
153
154     /*
155      * This instantiated message can now be given to the simulator. Note
156      * that here the simulator can potentially spot a failure, namely
157      * when this message is not specified in the SSM! In that sense,
158      * here we do testing.
159      */
160     aMbItIoNsims.processInstantiatedMessage(aMbItIoNim);
161
162     /*
163      * Ok, the simulator knows the input. Now we need a functionally
164      * correct response to this call. We first ask the simulator for all
165      * currently activated output transitions.
166      */
167     java.util.ArrayList aMbItIoNoutputs = new java.util.ArrayList(aMbItIoNsims.getCurrentOutputSwitches());
168
169     /*
170      * Out of all possible output switches, we randomly choose one and
171      * check if it has a solution. If yes, we take it. If not, we choose
172      * randomly the next one.
173      */
174     boolean aMbItIoNnoSolutionFound = true;
175     info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNnextOutput = null;
176     java.util.Random aMbItIoNrandom = new java.util.Random();
177     while (!aMbItIoNoutputs.isEmpty() && aMbItIoNnoSolutionFound) {
178         info.frantzen.testing.ssmsimulator.ssm.Switch aMbItIoNcandidate = (info.frantzen.testing.
179             ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));
180
181         /*
182          * try to find a solution, if yes, fine, if not, remove the
183          * candidate
184          */
185         aMbItIoNnextOutput = aMbItIoNsims.findSolution(aMbItIoNcandidate);
186         if (aMbItIoNnextOutput == null)
187             aMbItIoNoutputs.remove(aMbItIoNcandidate);
188         else
189             aMbItIoNnoSolutionFound = false;
190     }

```

```

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)
196                                     aMbItIoNreturnMessage.getType().iterator().next()).getName();
197     info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
198     getValuation();
199     info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
200     .getSingleInstance(aMbItIoNreturnVarName);
201     String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
202     aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue()
203     ),aMbItIoNarrayRepresentation[1],(Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()),(
204     Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
205     /*
206      * Now send the returnValue back to the calling service. That's it.
207      */
208 } catch (Exception genericException) {
209     throw new java.rmi.RemoteException(genericException.getMessage());
210 }
211 Density D = new Density();
212 Double sleepValue = D
213     .gaussian(25000 - puppet.ambition.Naturals
214     .asNatural(aMbItIoNinvocationTime
215     - System.currentTimeMillis()));
216 try {
217     Thread.sleep(sleepValue.longValue());
218 } catch (InterruptedException e) {
219 }
220 return aMbItIoNreturnValue;
221 }
222
223 public void cancelTransact(int ref) throws java.rmi.RemoteException {
224     long aMbItIoNinvocationTime = 0;
225     try {
226         aMbItIoNinvocationTime = System.currentTimeMillis();
227         /*
228          * Code Generated for Integration with Ambition
229          */
230         info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
231             ssmsimulator.ssm.Operation(
232                 "cancelTransact",
233                 info.frantzen.testing.ssmsimulator.ssm.OperationKind.ONeway);
234         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmmessage = aMbItIoNfindSSMMessag
235             e(aMbItIoNssm,
236             info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT,
237             aMbItIoNoperation);
238         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
239             ssmsimulator.ssm.Valuation();
240         java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmmessageType =
241             aMbItIoNmmessage
242             .getType();
243         java.util.Iterator aMbItIoNit = aMbItIoNmmessageType.iterator();
244         info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
245
246         /*
247          * Generated Parameter 0
248          */
249         aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit
250             .next();
251         info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
252             ssmsimulator.types.ST_PosIntInstance(
253                 ref);
254         aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),
255                 refInstance);
256
257         /*
258          * The valuation is ready, we can construct an instantiated message
259          */
260         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
261             ssmsimulator.ssm.InstantiatedMessage(
262                 aMbItIoNmmessage, aMbItIoNvaluation);

```

```

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.
280             ssmsimulator.ssm.Operation("orderShipment", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
281             ONEWAY);
282         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMesssage( aMbItIoNssm,
283             info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
284         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
285             ssmsimulator.ssm.Valuation();
286         java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmesssageType =
287             aMbItIoNmessage.getType();
288         java.util.Iterator aMbItIoNit = aMbItIoNmesssageType.iterator();
289         info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;
290
291         /*
292          * Generated Parameter 0
293          */
294         aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
295         info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
296             ssmsimulator.types.ST_PosIntInstance(ref);
297         aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), refInstance);
298
299         /*
300          * Generated Parameter 1
301          */
302         aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
303         Object[] aMbItIoNparameterValues = new Object[2];
304         aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
305             getFirstName());
306         aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
307             getLastname());
308         info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
309             ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
310         aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), adrInstance);
311
312         /*
313          * The valuation is ready, we can construct an instantiated message
314          */
315         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
316             ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);
317
318         /*
319          * This instantiated message can now be given to the simulator. Note
320          * that here the simulator can potentially spot a failure, namely
321          * when this message is not specified in the SSM! In that sense,
322          * here we do testing.
323          */
324         aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
325     } catch (Exception genericException) {
326         throw new java.rmi.RemoteException(genericException.getMessage());
327     }
328     long winSize = 120000;
329     int maxFault = 3;
330     long currentTimeStamp = System.currentTimeMillis();
331     for (int i=0; i<faultBuffer.size();i++){
332         if (currentTimeStamp - faultBuffer.get(i) >= winSize){
333             faultBuffer.remove(i);
334         }
335     }
336     if (faultBuffer.size() < maxFault){
337         Density d = new Density();
338     }
339 }
```

```

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] The network simulator NS-2 homepage. <http://www.isi.edu/nsnam/ns/>.
- [2] 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.
- [3] 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.
- [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 D4.1, PLASTIC Consortium, March 2007. IST STREP Project.
- [5] 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.
- [6] A. Bertolino, G. De Angelis, L. Frantzen, and A. Polini. Model-based Generation of Testbeds for Web Services. In *Proc. of the 20th IFIP Int. Conference on Testing of Communicating Systems (TESTCOM 2008)*, volume 5047 of LNCS, pages 266–282, Tokio, Japan, June 2008. Springer Verlag.
- [7] A. Bertolino, G. De Angelis, F. Lonetti, and A. Sabetta. Let the puppets move! automated testbed generation for service-oriented mobile applications. In *Proc. of the 34th EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (€μ-SEAA 2008)*, Parma, Italy, September 2008. IEEE. – to appear.
- [8] 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.
- [9] 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.
- [10] F. Liopoulos, 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.
- [11] H. Ludwig. WS-Agreement Concepts and Use - Agreement-Based Service-Oriented Architectures. Technical Report RC23949, IBM, May 2006.
- [12] J. Skene and W. Emmerich. Engineering runtime requirements: monitoring systems using MDA technologies. 2005.