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

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 provided good QoS values but the responses were not built to be semantically meaningful (i.e. always returning constant values). However, in the general case extra-functional aspects are tightly coupled with functional characteristics. The current version of PUPPET integrates the emulation of the functional specifications as part of the generated testbed. The obtained environment can expose not only the specified extra-functional parameters but also meaningful functional behavior. Specifically, PUPPET generates stubs for Web Services which respect both an extra-functional contract expressed via a Service Level Agreement (SLA), and a functional contract modeled via a Service State Machine (SSM, see Chapter 4.7).

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 to launch PUPPET :

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

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

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

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

Jambition Libs : SSMSimulator.jar, minerva.jar (see `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 correctely 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 exacutable 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 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>.

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

```
<wsag:AgreementOffer xsi:schemaLocation="..." xmlns:wsag="...">
...
    <wsag:GuaranteeTerm wsag:Name="WarehouseGT"
     wsag:Obligated="ServiceProvider">
     ...
    </wsag:GuaranteeTerm>
...
</wsag:AgreementOffer>
```

```
<tns:TrueGTList xmlns:tns="..." xmlns:xsi="..." xsi:schemaLocation="...">
...
 <tns:GTItemName>WarehouseGT</tns:GTItemName>
...
</tns:TrueGTList>
```

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.

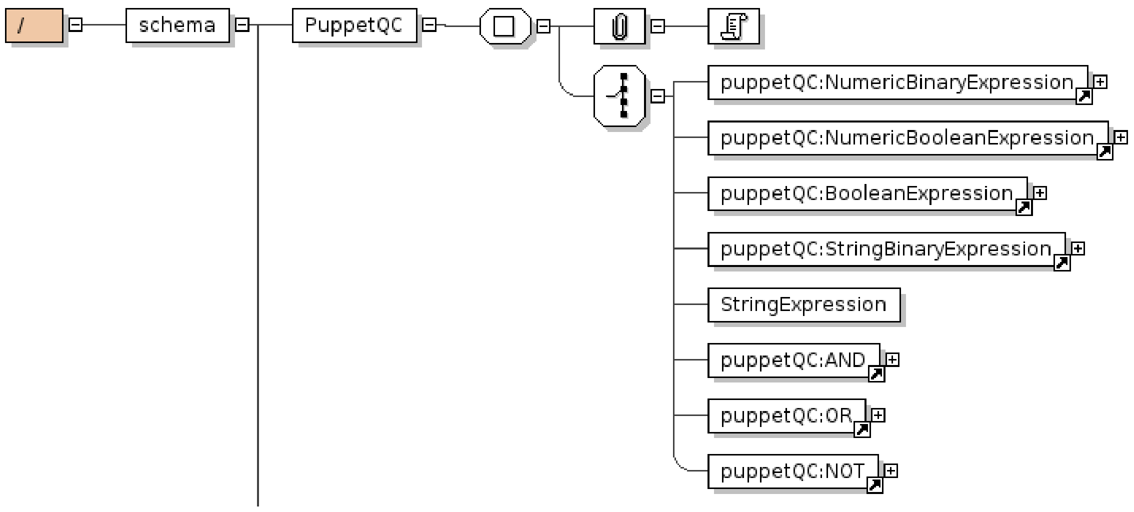

/
schema
PuppetQC
puppetQC:NumericBinaryExpression
puppetQC:NumericBooleanExpression
puppetQC:BooleanExpression
puppetQC:StringBinaryExpression
StringExpression
puppetQC:AND
puppetQC:OR
puppetQC:NOT

Figure 1: Expressions in the Qualifying Condition

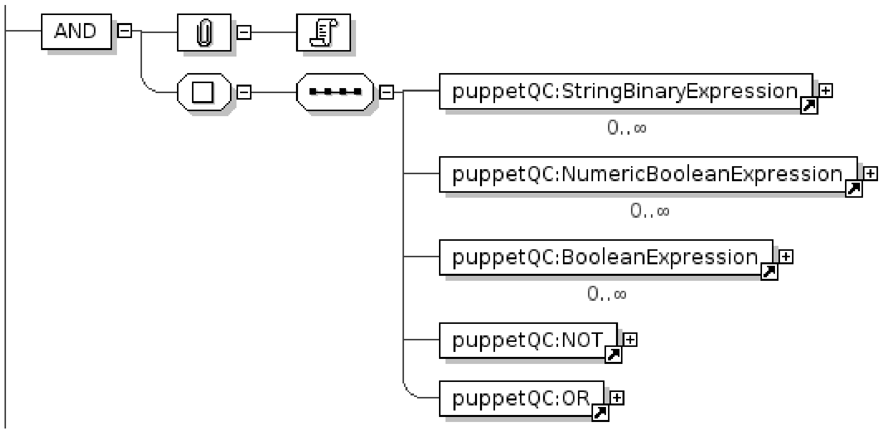

AND
puppetQC:StringBinaryExpression
0..∞
puppetQC:NumericBooleanExpression
0..∞
puppetQC:BooleanExpression
0..∞
puppetQC:NOT
puppetQC:OR

Figure 2: AND in the Qualifying Condition

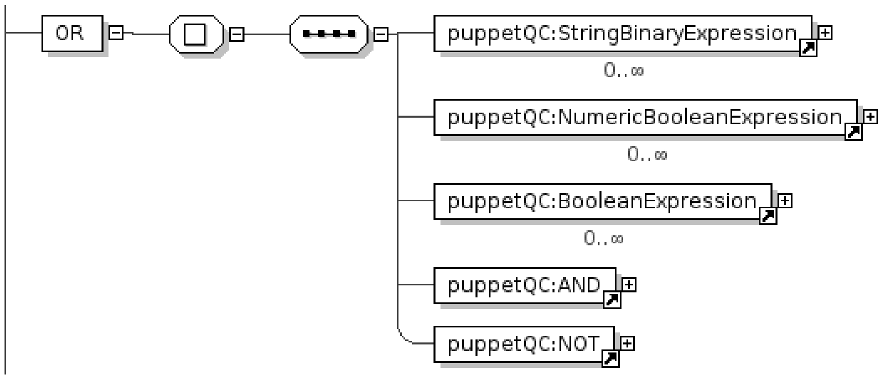

OR
puppetQC:StringBinaryExpression
0..∞
puppetQC:NumericBooleanExpression
0..∞
puppetQC:BooleanExpression
0..∞
puppetQC:AND
puppetQC:NOT

Figure 3: OR in the Qualifying Condition

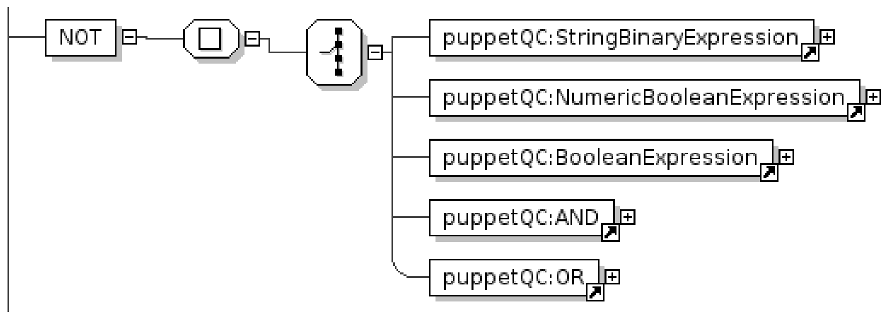

NOT
puppetQC:StringBinaryExpression
puppetQC:NumericBooleanExpression
puppetQC:BooleanExpression
puppetQC:AND
puppetQC:OR

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).

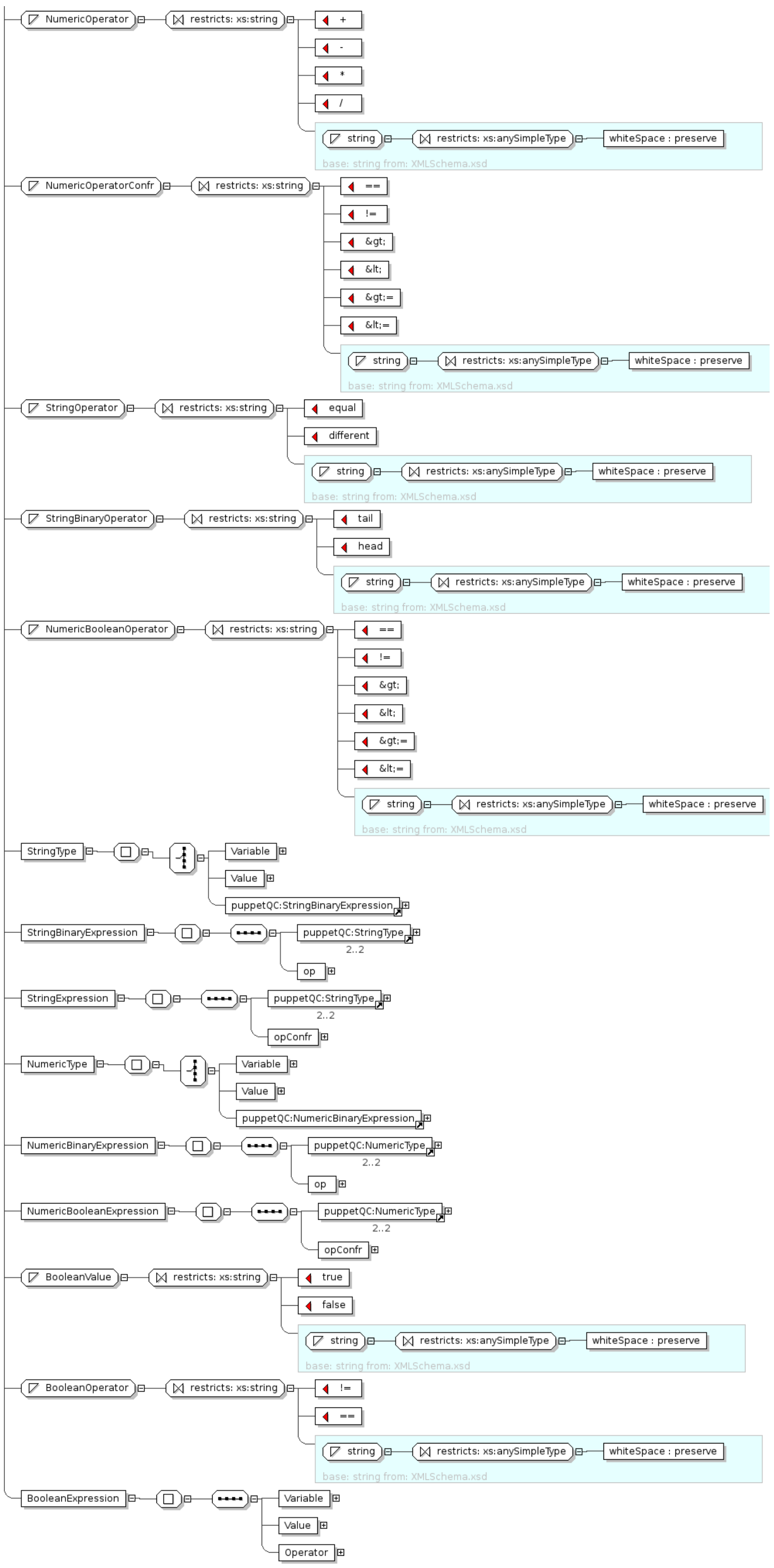


Figure 5: Operators in the Qualifying Condition

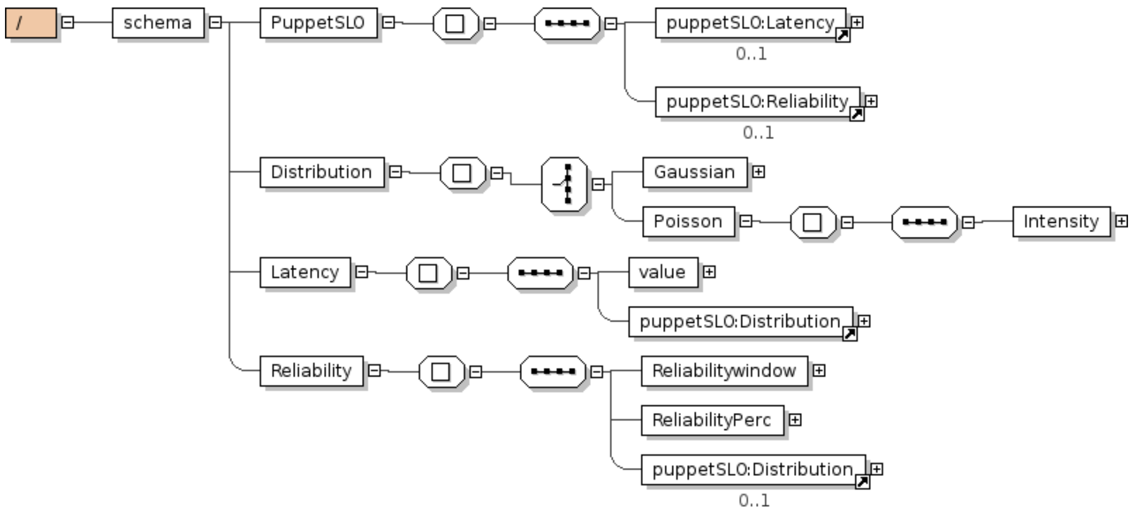


Figure 6: Extra-Functional Properties in the Service Level Objective

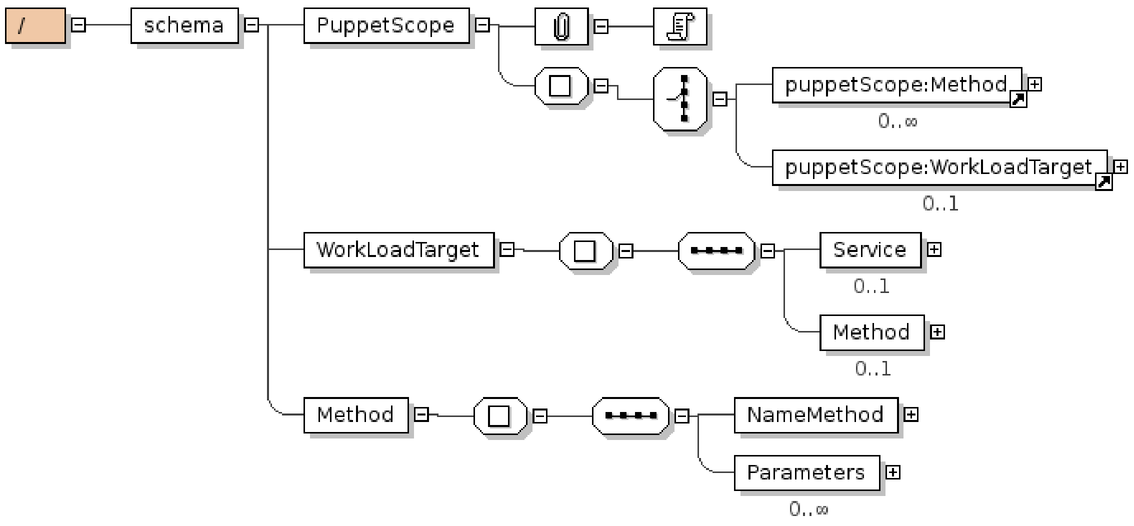


Figure 7: Defining the Scope of a Term

```
    /*
     * The SSM object.
     */
    private info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine aMbItIoNssm;

    /*
     * The simulator is generated
     */
    private info.frantzen.testing.ssmsimulator.SSMSimulator aMbItIoNsim;

    private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMessage(
            info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine ssm,
            info.frantzen.testing.ssmsimulator.ssm.MessageKind kind,
            info.frantzen.testing.ssmsimulator.ssm.Operation op)
            throws Exception {
        java.util.HashSet<info.frantzen.testing.ssmsimulator.ssm.Message> messages = ssm
                .getMessages();
        for (java.util.Iterator it = messages.iterator(); it.hasNext();) {
            info.frantzen.testing.ssmsimulator.ssm.Message m = (info.frantzen.testing.ssmsimulator.ssm.Message) it
                    .next();
            if (m.getKind() != info.frantzen.testing.ssmsimulator.ssm.MessageKind.UNOBSERVABLE) {
                if ((m.getKind() == kind) && (m.getOperation().equals(op))) {
                    return m;
                }
            }
        }
        throw new Exception(
                "Cannot␣find␣the␣input␣SSM␣message␣belonging␣to␣the␣Operation␣"
                        + op.getName() + "!");
    }
```

Listing 1: Attributes and Local operation included in the code

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

- at line 49 set the URL of the WSDL the Web Service that is going to be emulated exports
- at line 53 set the name of the Web Service that is going to be emulated as reported on the WSDL
- at line 57 set the port of the Web Service that is going to be emulated as reported on the WSDL
- at line 61 set the URL of the file with the specification of the SSM
- at line 76 set the URL of the treeSolver that the simulator uses in order to generate meaningful functional values
- at line 77 set the port of the treeSolver that the simulator uses in order to generate meaningful functional values

```
42          /*
43           * To initialise the Simulator, the following items are needed:
44           */
45
46          /*
47           * The URL of the WSDL file
48           */
49          java.net.URL aMbItIoNWSDLUrl = new java.net.URL("Put␣here␣the␣URL␣of␣the␣Service's␣WSDL");
50          /*
51           * The name of the WSDL-Service
52           */
53          String aMbItIoNservice = "Put␣here␣the␣name␣of␣the␣Service␣as␣in␣the␣WSDL";
54          /*
55           * The name of the WSDL-Port
56           */
57          String aMbItIoNport = "Put␣here␣the␣port␣of␣the␣Service␣as␣in␣the␣WSDL";
58          /*
59           * The URL of the SSM Schema Instance
60           */
61          java.net.URL aMbItIoNSSMUrl = new java.net.URL("Put␣here␣the␣URL␣of␣the␣SSM␣Schema␣Instance");
62
63          /*
64           * Now we can generate the SSM object. To do so, we use Zsolt's "Minerva" library
65           */
66          aMbItIoNssm = hu.soft4d.jessi.ssm.SSMHandler.generateSSM(
```

```java
            aMbItIoNWSDLUrl, aMbItIoNSSMUrl, aMbItIoNservice, aMbItIoNport);
        /*
         * Before we can use the SSM in the simulator, the parsers have to be attached
         * to the switches
         */
        aMbItIoNssm.attachParsersToSwitches();
        /*
         * Next we generate the socket to the treeSolver.
         */
        String aMbItIoNsolverHost = "Put␣here␣the␣URL␣of␣the␣Solver";
        int aMbItIoNsolverPort = "Put␣here␣the␣Port␣the␣Solver";
        java.net.Socket aMbItIoNsolverSocket = new java.net.Socket(
            aMbItIoNsolverHost, aMbItIoNsolverPort);
        /*
         * The treeSolver sends a welcome message, we remove it from the stream
         */
        new java.io.BufferedReader(new java.io.InputStreamReader(
            aMbItIoNsolverSocket.getInputStream())).readLine();
        /*
         * The simulator can use an external tool to display sequence diagrams
         * of the messages exchanged. // I skip this here since this takes extra
         * ressources/
         */

        /*
         * The simulator needs a logger to log to
         */
        java.util.logging.Logger aMbItIoNlogger = java.util.logging.Logger
            .getLogger("");

        /* The simulator is generated */

        aMbItIoNsim = new info.frantzen.testing.ssmsimulator.SSMSimulator(
            aMbItIoNssm, aMbItIoNsolverSocket, aMbItIoNlogger);

        /*
         * If Double variables are used we assume this models money
         * (experimental). In any case, do this:
         */
        info.frantzen.testing.ssmsimulator.types.ST_PseudoPosDouble.postPointLength = 2;
        /*
         * Now the Simulator is ready.
         * -------------------------------------------------------------------------
         */
```

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.

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

Listing 3: Local Variables Configuration

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

```java
            /*
             * Generated Parameter 0
             */
            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
            info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
                ssmsimulator.types.ST_PosIntInstance(ref);
```

```
        aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), refInstance);

        /*
         * Generated Parameter 1
         */
        aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
        Object[] aMbItIoNparameterValues = new Object[2];
        aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
            getFirstName());
        aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
            getLastName());
        info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
            ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
        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.

```
        /*
         * The valuation is ready, we can construct an instantiated message
         */
        info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
            ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);

        /*
         * This instantiated message can now be given to the simulator. Note
         * that here the simulator can potentially spot a failure, namely
         * when this message is not specified in the SSM! In that sense,
         * here we do testing.
         */
        aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
    } catch (Exception genericException) {
        throw new java.rmi.RemoteException(genericException.getMessage());
    }
```

Listing 5: The Generation of the Operation's Parameters

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

```
153         /*
154          * Ok, the simulator knows the input. Now we need a functionally
155          * correct response to this call. We first ask the simulator for all
156          * currently activated output transitions.
157          */
158         java.util.ArrayList aMbItIoNoutputs = new java.util.ArrayList(aMbItIoNsim.getCurrentOutputSwitches());
159 
160         /*
161          * Out of all possible output switches, we randomly choose one and
162          * check if it has a solution. If yes, we take it. If not, we choose
163          * randomly the next one.
164          */
165         boolean aMbItIoNnoSolutionFound = true;
166         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNnextOutput = null;
167         java.util.Random aMbItIoNrandom = new java.util.Random();
168         while (!aMbItIoNoutputs.isEmpty() && aMbItIoNnoSolutionFound) {
169            info.frantzen.testing.ssmsimulator.ssm.Switch aMbItIoNcandidate = (info.frantzen.testing.
                   ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));
170 
171            /*
172             * try to find a solution, if yes, fine, if not, remove the
173             * candidate
174             */
175            aMbItIoNnextOutput = aMbItIoNsim.findSolution(aMbItIoNcandidate);
176            if (aMbItIoNnextOutput == null)
177               aMbItIoNoutputs.remove(aMbItIoNcandidate);
178            else
179               aMbItIoNnoSolutionFound = false;
180         }
181         if (aMbItIoNnextOutput == null)
182            throw new Exception("Failure␣in␣SSM!␣No␣output␣for␣synchronous␣input␣specified!");
183         /*
184          * Ok, we have now a feasible and functionally correct output:
185          * nextOutput Before we send this output to the Service out there,
186          * we tell so to the simulator:
187          */
188         aMbItIoNsim.processInstantiatedMessageNoBackup(aMbItIoNnextOutput);
189
190         /*
191          * What is left to do, is to map this instantiated message back to a
192          * real returnValue.
193          */
194         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNreturnMessage = aMbItIoNnextOutput.getMessage()
                 ;
195         String aMbItIoNreturnVarName = ((info.frantzen.testing.ssmsimulator.ssm.InteractionVariable)
                 aMbItIoNreturnMessage.getType().iterator().next()).getName();
196         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
                 getValuation();
197         info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
                 .getSingleInstance(aMbItIoNreturnVarName);
198         String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
199         aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue()
                 ),aMbItIoNarrayRepresentation[1],(Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()),(
                 Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
200         /*
201          * Now send the returnValue back to the calling service. That's it.
202          */
```

Listing 6: The Generation of the Meaningful Return Value

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

For the sake of completeness, in the appendix is reported the Java source code emulating a Warehouse Web Service (see the example in Chapter 4.7). 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 definded by means of NS-2's `setdest` tool. The results of an NS-2 simulations with the movement specifications can 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 service 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 requiremnets of each scenario.

PUPPET includes a defaul 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 inuencing 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 Congifuration of the Mobility Oracle

The Mobility Oracle (implemented in `puppet.jar` as `AbstractTraceReader` and `TraceReader`) can be configured by meas 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 movment 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 describe the mobility of the nodes in the scenario. Bu 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 feature that it exports. Specifically the operation **notifyAlarm** is aimed at both collect and process the alarm messages coming from the eHealth part of the application that runs on Plastic. It is invoked when an alarm is raised. It takes as input the name of the Residential Gateway where the alarm comes from, the kind of the alarm and the date when it was raised.

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

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

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

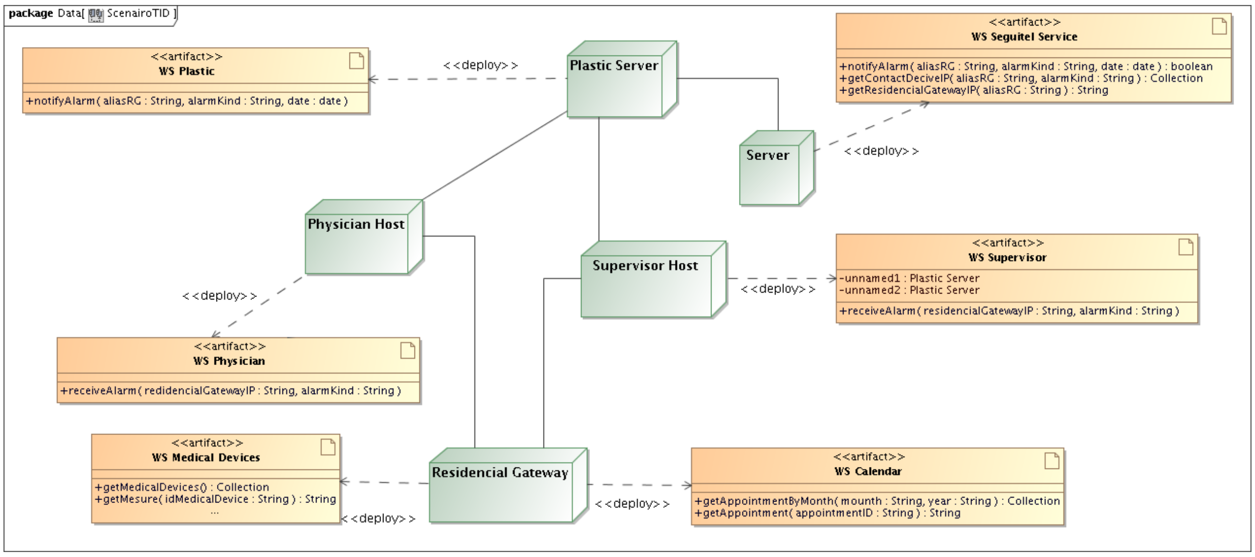

package Data[ScenairoTID]
<<artifact>>
WS Plastic
+notifyAlarm(aliasRG : String, alarmKind : String, date : date)
<<deploy>>
Plastic Server
<<artifact>>
WS Seguitel Service
+notifyAlarm(aliasRG : String, alarmKind : String, date : date) : boolean
+getContactDeciveIP(aliasRG : String, alarmKind : String) : Collection
+getResidencialGatewayIP(aliasRG : String) : String
Server
<<deploy>>
Physician Host
<<deploy>>
<<artifact>>
WS Physician
+receiveAlarm(redidencialGatewayIP : String, alarmKind : String)
Supervisor Host
<<deploy>>
<<artifact>>
WS Supervisor
-unnamed1 : Plastic Server
-unnamed2 : Plastic Server
+receiveAlarm(residencialGatewayIP : String, alarmKind : String)
<<artifact>>
WS Medical Devices
+getMedicalDevices() : Collection
+getMesure(idMedicalDevice : String) : String
...
<<deploy>>
Residencial Gateway
<<deploy>>
<<artifact>>
WS Calendar
+getAppointmentByMonth(mounth : String, year : String) : Collection
+getAppointment(appointmentID : String) : String

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.

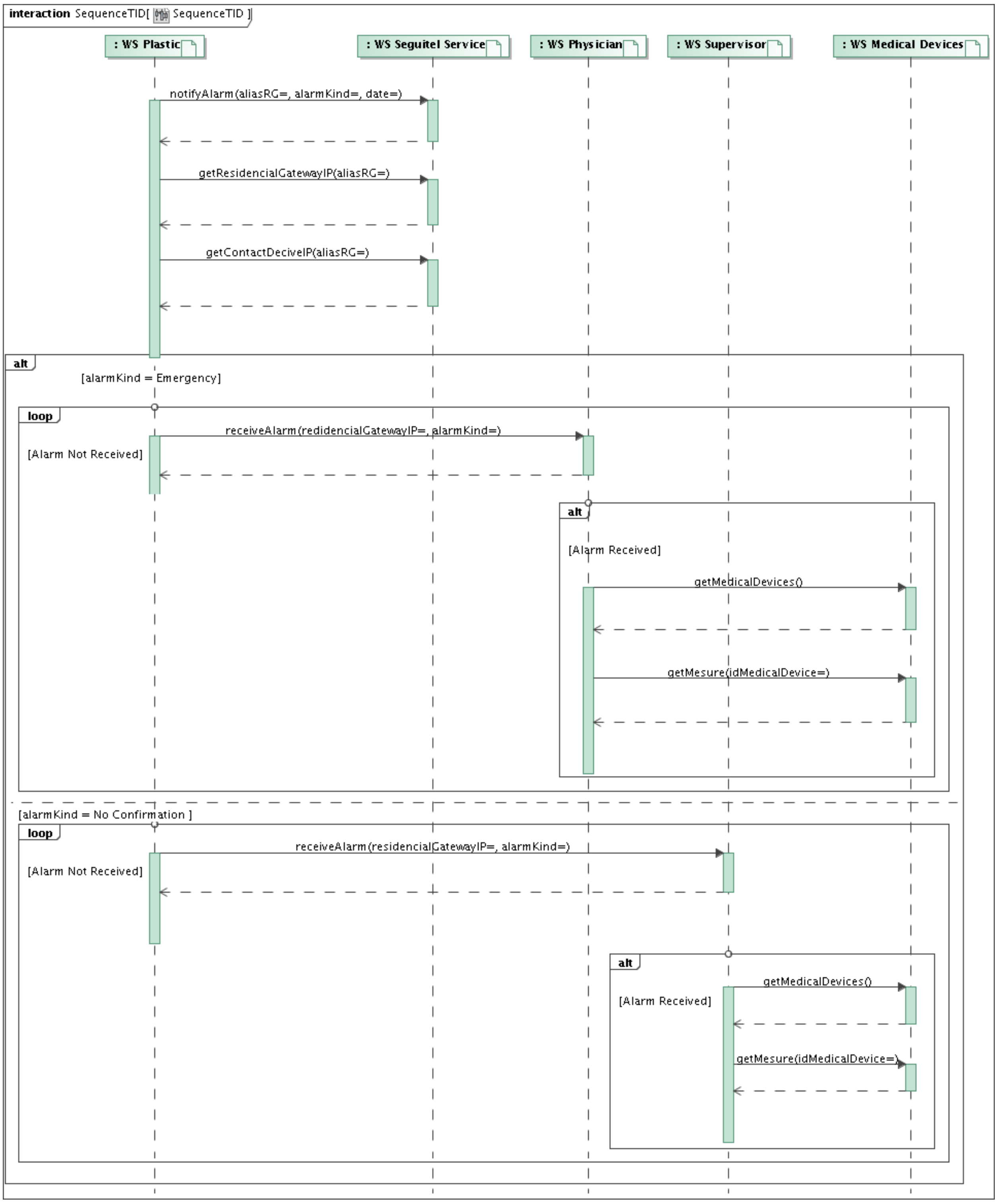
As for the doctor, also here the hosting device could be either a usual wired device as a PC or a mobile and wireless device such as a smart phone. The Web Service exports the **receiveAlarm** operation. Such operation takes as input both the IP address of the Residential Gateway where the alarm was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



interaction SequenceTID[SequenceTID]
: WS Plastic
: WS Seguitel Service
: WS Physician
: WS Supervisor
: WS Medical Devices
notifyAlarm(aliasRG=, alarmKind=, date=)
getResidencialGatewayIP(aliasRG=)
getContactDeciveIP(aliasRG=)
alt
[alarmKind = Emergency]
loop
receiveAlarm(redidencialGatewayIP=, alarmKind=)
[Alarm Not Received]
alt
[Alarm Received]
getMedicalDevices()
getMesure(idMedicalDevice=)
[alarmKind = No Confirmation]
loop
receiveAlarm(residencialGatewayIP=, alarmKind=)
[Alarm Not Received]
alt
[Alarm Received]
getMedicalDevices()
getMesure(idMedicalDevice=)

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

Table 2: QoS Properties

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

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

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

4.7 Reference To Jambition

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

5 Appendix

5.1 Abstract WSDL: WSDoctor

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSDoctor" xmlns:apachesoap="http://xml.
     apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSDoctor" xmlns:intf="http://
     localhost:8080/axis/services/WSDoctor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl
     ="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http:
     //www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->

  <wsdl:message name="receiveAlarmRequest">

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

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

  </wsdl:message>

  <wsdl:message name="receiveAlarmResponse">

  </wsdl:message>

  <wsdl:portType name="WSDoctor">

    <wsdl:operation name="receiveAlarm" parameterOrder="residencialGatewayIP␣alarmGender">

      <wsdl:input message="impl:receiveAlarmRequest" name="receiveAlarmRequest"/>

      <wsdl:output message="impl:receiveAlarmResponse" name="receiveAlarmResponse"/>

    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="WSDoctorSoapBinding" type="impl:WSDoctor">

    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="receiveAlarm">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="receiveAlarmRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
             use="encoded"/>

      </wsdl:input>

      <wsdl:output name="receiveAlarmResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
            localhost:8080/axis/services/WSDoctor" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

  </wsdl:binding>

  <wsdl:service name="WSDoctorService">
```

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

```

    <wsdl:port binding="impl:WSDoctorSoapBinding" name="WSDoctor">

      <wsdlsoap:address location="http://localhost:8080/axis/services/WSDoctor"/>

    </wsdl:port>

  </wsdl:service>

</wsdl:definitions>
```

5.2 Abstract WSDL: WSCalendar

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSCalendar" xmlns:apachesoap="http://xml
     .apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSCalendar" xmlns:intf="http://
     localhost:8080/axis/services/WSCalendar" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema targetNamespace="http://localhost:8080/axis/services/WSCalendar" xmlns="http://www.w3.org/2001/
       XMLSchema">
   <import namespace="http://xml.apache.org/xml-soap"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_xsd_anyType">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
     </restriction>
    </complexContent>
   </complexType>
  </schema>
  <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://localhost:8080/axis/services/WSCalendar"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="Vector">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
    </sequence>
   </complexType>
  </schema>
 </wsdl:types>

   <wsdl:message name="getAppointmentByMonthRequest">

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

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

   </wsdl:message>

   <wsdl:message name="getAppointmentRequest">

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

   </wsdl:message>

   <wsdl:message name="getAppointmentByMonthResponse">

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

   </wsdl:message>

   <wsdl:message name="getAppointmentResponse">

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

   </wsdl:message>

   <wsdl:portType name="WSCalendar">

      <wsdl:operation name="getAppointmentByMonth" parameterOrder="month year">

         <wsdl:input message="impl:getAppointmentByMonthRequest" name="getAppointmentByMonthRequest"/>

         <wsdl:output message="impl:getAppointmentByMonthResponse" name="getAppointmentByMonthResponse"/>

      </wsdl:operation>

      <wsdl:operation name="getAppointment" parameterOrder="apointmentID">

```

```
        <wsdl:input message="impl:getAppointmentRequest" name="getAppointmentRequest"/>

        <wsdl:output message="impl:getAppointmentResponse" name="getAppointmentResponse"/>

      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="WSCalendarSoapBinding" type="impl:WSCalendar">

      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="getAppointmentByMonth">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="getAppointmentByMonthRequest">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                 use="encoded"/>

         </wsdl:input>

         <wsdl:output name="getAppointmentByMonthResponse">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSCalendar" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getAppointment">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="getAppointmentRequest">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
                 use="encoded"/>

         </wsdl:input>

         <wsdl:output name="getAppointmentResponse">

            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
                localhost:8080/axis/services/WSCalendar" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="WSCalendarService">

      <wsdl:port binding="impl:WSCalendarSoapBinding" name="WSCalendar">

         <wsdlsoap:address location="http://localhost:8080/axis/services/WSCalendar"/>

      </wsdl:port>

   </wsdl:service>

</wsdl:definitions>
```

5.3 Abstract WSDL: WSSupervisor

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSSupervisor" xmlns:apachesoap="http://
     xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSupervisor" xmlns:intf="http://
     localhost:8080/axis/services/WSSupervisor" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->

   <wsdl:message name="receiveAlarmResponse">

   </wsdl:message>

   <wsdl:message name="receiveAlarmRequest">

```

```
    <wsdl:part name="residencialGatewayIP" type="soapenc:string"/>

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

  </wsdl:message>

  <wsdl:portType name="WSSupervisor">

    <wsdl:operation name="receiveAlarm" parameterOrder="residencialGatewayIP␣alarmGender">

      <wsdl:input message="impl:receiveAlarmRequest" name="receiveAlarmRequest"/>

      <wsdl:output message="impl:receiveAlarmResponse" name="receiveAlarmResponse"/>

    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="WSSupervisorSoapBinding" type="impl:WSSupervisor">

    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="receiveAlarm">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="receiveAlarmRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
             use="encoded"/>

      </wsdl:input>

      <wsdl:output name="receiveAlarmResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
            localhost:8080/axis/services/WSSupervisor" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

  </wsdl:binding>

  <wsdl:service name="WSSupervisorService">

    <wsdl:port binding="impl:WSSupervisorSoapBinding" name="WSSupervisor">

      <wsdlsoap:address location="http://localhost:8080/axis/services/WSSupervisor"/>

    </wsdl:port>

  </wsdl:service>

</wsdl:definitions>
```

5.4 Abstract WSDL: WSPlastic

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->

  <wsdl:message name="notifyAlarmRequest">

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

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

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

  </wsdl:message>

  <wsdl:message name="notifyAlarmResponse">

  </wsdl:message>

  <wsdl:portType name="WSPlastic">

```

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

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

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

    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="WSPlasticSoapBinding" type="impl:WSPlastic">

    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="notifyAlarm">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="notifyAlarmRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
             use="encoded"/>

      </wsdl:input>

      <wsdl:output name="notifyAlarmResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
            localhost:8080/axis/services/WSPlastic" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

  </wsdl:binding>

  <wsdl:service name="WSPlasticService">

    <wsdl:port binding="impl:WSPlasticSoapBinding" name="WSPlastic">

      <wsdlsoap:address location="http://localhost:8080/axis/services/WSPlastic"/>

    </wsdl:port>

  </wsdl:service>

</wsdl:definitions>
```

5.5 Abstract WSDL: WSSeguitelService

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/services/WSSeguitelService" xmlns:apachesoap="
     http://xml.apache.org/xml-soap" xmlns:impl="http://localhost:8080/axis/services/WSSeguitelService"
     xmlns:intf="http://localhost:8080/axis/services/WSSeguitelService" xmlns:soapenc="http://schemas.xmlsoap.
     org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.
     org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema targetNamespace="http://localhost:8080/axis/services/WSSeguitelService" xmlns="http://www.w3.org/2001/
       XMLSchema">
   <import namespace="http://xml.apache.org/xml-soap"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_xsd_anyType">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
     </restriction>
    </complexContent>
   </complexType>
  </schema>
  <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://localhost:8080/axis/services/WSSeguitelService"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="Vector">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
    </sequence>
   </complexType>
  </schema>
 </wsdl:types>

   <wsdl:message name="getResidencialGatewayIPResponse">
```

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

  </wsdl:message>

  <wsdl:message name="getResidencialGatewayIPRequest">

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

  </wsdl:message>

  <wsdl:message name="notifyAlarmRequest">

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

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

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

  </wsdl:message>

  <wsdl:message name="getConnectedDeviceIPRequest">

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

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

  </wsdl:message>

  <wsdl:message name="notifyAlarmResponse">

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

  </wsdl:message>

  <wsdl:message name="getConnectedDeviceIPResponse">

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

  </wsdl:message>

  <wsdl:portType name="WSSeguitelService">

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

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

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

    </wsdl:operation>

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

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

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

    </wsdl:operation>

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

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

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

    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="WSSeguitelServiceSoapBinding" type="impl:WSSeguitelService">

    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="notifyAlarm">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="notifyAlarmRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
             use="encoded"/>

      </wsdl:input>

      <wsdl:output name="notifyAlarmResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSSeguitelService" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

    <wsdl:operation name="getConnectedDeviceIP">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="getConnectedDeviceIPRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
              use="encoded"/>

      </wsdl:input>

      <wsdl:output name="getConnectedDeviceIPResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSSeguitelService" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

    <wsdl:operation name="getResidencialGatewayIP">

      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="getResidencialGatewayIPRequest">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
              use="encoded"/>

      </wsdl:input>

      <wsdl:output name="getResidencialGatewayIPResponse">

        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSSeguitelService" use="encoded"/>

      </wsdl:output>

    </wsdl:operation>

  </wsdl:binding>

  <wsdl:service name="WSSeguitelServiceService">

    <wsdl:port binding="impl:WSSeguitelServiceSoapBinding" name="WSSeguitelService">

      <wsdlsoap:address location="http://localhost:8080/axis/services/WSSeguitelService"/>

    </wsdl:port>

  </wsdl:service>

</wsdl:definitions>
```

5.6 Abstract WSDL: WSMedicalDevice

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema targetNamespace="http://localhost:8080/axis/services/WSMedicalDevice" xmlns="http://www.w3.org/2001/
       XMLSchema">
   <import namespace="http://xml.apache.org/xml-soap"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_xsd_anyType">
    <complexContent>
     <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
     </restriction>
    </complexContent>
   </complexType>
  </schema>
  <schema targetNamespace="http://xml.apache.org/xml-soap" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://localhost:8080/axis/services/WSMedicalDevice"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="Vector">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
    </sequence>
   </complexType>
  </schema>
 </wsdl:types>

   <wsdl:message name="getMedicalDevicesRequest">

   </wsdl:message>

   <wsdl:message name="getMedicalDevicesResponse">

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

   </wsdl:message>

   <wsdl:message name="getMeasureRequest">

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

   </wsdl:message>

   <wsdl:message name="getMeasureResponse">

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

   </wsdl:message>

   <wsdl:portType name="WSMedicalDevice">

     <wsdl:operation name="getMedicalDevices">

       <wsdl:input message="impl:getMedicalDevicesRequest" name="getMedicalDevicesRequest"/>

       <wsdl:output message="impl:getMedicalDevicesResponse" name="getMedicalDevicesResponse"/>

     </wsdl:operation>

     <wsdl:operation name="getMeasure" parameterOrder="idMedicalDevice">

       <wsdl:input message="impl:getMeasureRequest" name="getMeasureRequest"/>

       <wsdl:output message="impl:getMeasureResponse" name="getMeasureResponse"/>

     </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="WSMedicalDeviceSoapBinding" type="impl:WSMedicalDevice">

     <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

     <wsdl:operation name="getMedicalDevices">

       <wsdlsoap:operation soapAction=""/>

       <wsdl:input name="getMedicalDevicesRequest">

         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
              use="encoded"/>

       </wsdl:input>

       <wsdl:output name="getMedicalDevicesResponse">

         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSMedicalDevice" use="encoded"/>

       </wsdl:output>

     </wsdl:operation>

     <wsdl:operation name="getMeasure">

       <wsdlsoap:operation soapAction=""/>

       <wsdl:input name="getMeasureRequest">

         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://wsScenario"
              use="encoded"/>

       </wsdl:input>

       <wsdl:output name="getMeasureResponse">

         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
             localhost:8080/axis/services/WSMedicalDevice" use="encoded"/>

       </wsdl:output>

     </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="WSMedicalDeviceService">

     <wsdl:port binding="impl:WSMedicalDeviceSoapBinding" name="WSMedicalDevice">

       <wsdlsoap:address location="http://localhost:8080/axis/services/WSMedicalDevice"/>

     </wsdl:port>

   </wsdl:service>

</wsdl:definitions>
```

5.7 WS Agreement

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:AgreementOffer xsi:schemaLocation="http://www.ggf.org/namespaces/ws-agreement"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xs="http://www.w3.org/2001/XMLSchema"
               xmlns:wsag="http://schemas.ggf.org/graap/2005/09/ws-agreement"
               xmlns:puppetScope="http://setest0.isti.cnr.it/puppetScope"
               xmlns:puppetSLO="http://setest0.isti.cnr.it/puppetSLO"
               xmlns:puppetQC="http://setest0.isti.cnr.it/puppetQC"
               xmlns:ns="http://setest0.isti.cnr.it/puppet">
  <wsag:Name>Telefonica_Example_Scenario_3</wsag:Name>

  <wsag:Context />

  <wsag:Terms>
    <wsag:All>
      <wsag:GuaranteeTerm wsag:Name="ContactDeciveIP-Term1"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="WSSeguitelService">
          <puppetScope:PuppetScope>
            <puppetScope:Method>
<!--            <NameMethod>getContactDeciveIP</NameMethod> -->
              <NameMethod>getConnectedDeviceIP</NameMethod>
            </puppetScope:Method>
          </puppetScope:PuppetScope>
        </wsag:ServiceScope>

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

              <op>equal</op>

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

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

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

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

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

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

              <op>equal</op>

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

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

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

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

            <wsag:ValueExpression> 2 </wsag:ValueExpression>
          </wsag:Penalty>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:ExactlyOne>
        <wsag:GuaranteeTerm wsag:Name="AlarmDoctor-Term1"
          wsag:Obligated="ServiceProvider">
          <wsag:ServiceScope wsag:ServiceName="WSDoctor">
            <puppetScope:PuppetScope>
              <puppetScope:Method>
                <NameMethod>receiveAlarm</NameMethod>
              </puppetScope:Method>
            </puppetScope:PuppetScope>
          </wsag:ServiceScope>

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

                <op>equal</op>

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

          <wsag:ServiceLevelObjective>
            <puppetSLO:PuppetSLO>
              <puppetSLO:Latency>
                <value>6000</value>
                <puppetSLO:Distribution>
                    <Gaussian>10</Gaussian>
                </puppetSLO:Distribution>
              </puppetSLO:Latency>
              <puppetSLO:Reliability>
                <Reliabilitywindow>30000</Reliabilitywindow>
                <ReliabilityPerc>5</ReliabilityPerc>
                <puppetSLO:Distribution>
                  <Gaussian>100</Gaussian>
                </puppetSLO:Distribution>
              </puppetSLO:Reliability>
            </puppetSLO:PuppetSLO>
          </wsag:ServiceLevelObjective>

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

              <wsag:ValueExpression> 2 </wsag:ValueExpression>
            </wsag:Penalty>
          </wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
        <wsag:GuaranteeTerm wsag:Name="AlarmDoctor-Term2"
          wsag:Obligated="ServiceProvider">
          <wsag:ServiceScope wsag:ServiceName="WSDoctor">
            <puppetScope:PuppetScope>
              <puppetScope:Method>
                <NameMethod>receiveAlarm</NameMethod>
              </puppetScope:Method>
            </puppetScope:PuppetScope>
          </wsag:ServiceScope>

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

                <op>equal</op>

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

          <wsag:ServiceLevelObjective>
            <puppetSLO:PuppetSLO>
              <puppetSLO:Latency>
                <value>2000</value>
                  <puppetSLO:Distribution>
                    <Gaussian>10</Gaussian>
                  </puppetSLO:Distribution>
              </puppetSLO:Latency>
              <puppetSLO:Reliability>
                <Reliabilitywindow>30000</Reliabilitywindow>

                <ReliabilityPerc>1</ReliabilityPerc>

                <puppetSLO:Distribution>
                  <Gaussian>100</Gaussian>
                </puppetSLO:Distribution>
              </puppetSLO:Reliability>
            </puppetSLO:PuppetSLO>
          </wsag:ServiceLevelObjective>

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

```

            <wsag:ValueExpression> 2 </wsag:ValueExpression>
          </wsag:Penalty>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
    </wsag:ExactlyOne>
    <wsag:ExactlyOne>
      <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term1"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
          <puppetScope:PuppetScope>
            <puppetScope:Method>
              <NameMethod>receiveAlarm</NameMethod>
            </puppetScope:Method>
          </puppetScope:PuppetScope>
        </wsag:ServiceScope>

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

              <op>equal</op>

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

        <wsag:ServiceLevelObjective>
          <puppetSLO:PuppetSLO>
            <puppetSLO:Latency>
              <value>10000</value>

                <puppetSLO:Distribution>
                  <Gaussian>10</Gaussian>
                </puppetSLO:Distribution>
            </puppetSLO:Latency>
            <puppetSLO:Reliability>
              <Reliabilitywindow>30000</Reliabilitywindow>

              <ReliabilityPerc>5</ReliabilityPerc>

              <puppetSLO:Distribution>
                <Gaussian>100</Gaussian>
              </puppetSLO:Distribution>
            </puppetSLO:Reliability>
          </puppetSLO:PuppetSLO>
        </wsag:ServiceLevelObjective>

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

            <wsag:ValueExpression> 2 </wsag:ValueExpression>
          </wsag:Penalty>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:GuaranteeTerm wsag:Name="AlarmSupervisor-Term2"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="WSSupervisor">
          <puppetScope:PuppetScope>
            <puppetScope:Method>
              <NameMethod>receiveAlarm</NameMethod>
            </puppetScope:Method>
          </puppetScope:PuppetScope>
        </wsag:ServiceScope>

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

              <op>equal</op>

              <puppetQC:StringType>
```

```xml
            <Value>WiredServer</Value>
          </puppetQC:StringType>
        </puppetQC:StringBinaryExpression>
      </puppetQC:PuppetQC>
    </wsag:QualifyingCondition>

    <wsag:ServiceLevelObjective>
      <puppetSLO:PuppetSLO>
        <puppetSLO:Latency>
          <value>6000</value>

            <puppetSLO:Distribution>
              <Gaussian>10</Gaussian>
            </puppetSLO:Distribution>
        </puppetSLO:Latency>
        <puppetSLO:Reliability>
          <Reliabilitywindow>30000</Reliabilitywindow>

          <ReliabilityPerc>1</ReliabilityPerc>

          <puppetSLO:Distribution>
            <Gaussian>100</Gaussian>
          </puppetSLO:Distribution>
        </puppetSLO:Reliability>
      </puppetSLO:PuppetSLO>
    </wsag:ServiceLevelObjective>

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

        <wsag:ValueExpression> 2 </wsag:ValueExpression>
      </wsag:Penalty>
    </wsag:BusinessValueList>
  </wsag:GuaranteeTerm>
</wsag:ExactlyOne>
<wsag:GuaranteeTerm wsag:Name="MedicalDevice-Term1"
  wsag:Obligated="ServiceProvider">
  <wsag:ServiceScope wsag:ServiceName="WSMedicalDevice">
    <puppetScope:PuppetScope>
      <puppetScope:Method>
        <NameMethod>getMeasure</NameMethod>
      </puppetScope:Method>
    </puppetScope:PuppetScope>
  </wsag:ServiceScope>

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

        <op>equal</op>

        <puppetQC:StringType>
          <Value>device_1</Value>
        </puppetQC:StringType>
      </puppetQC:StringBinaryExpression>
    </puppetQC:PuppetQC>
  </wsag:QualifyingCondition>

  <wsag:ServiceLevelObjective>
    <puppetSLO:PuppetSLO>
      <puppetSLO:Latency>
        <value>3000</value>

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

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

      <wsag:ValueExpression> 2 </wsag:ValueExpression>
    </wsag:Penalty>
```

```
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:GuaranteeTerm wsag:Name="MedicalDevice-Term2"
        wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="WSMedicalDevice">
          <puppetScope:PuppetScope>
            <puppetScope:Method>
              <NameMethod>getMeasure</NameMethod>
            </puppetScope:Method>
          </puppetScope:PuppetScope>
        </wsag:ServiceScope>

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

              <op>equal</op>

              <puppetQC:StringType>
                <Value>device_2</Value>
              </puppetQC:StringType>
            </puppetQC:StringBinaryExpression>
          </puppetQC:PuppetQC>
        </wsag:QualifyingCondition>

        <wsag:ServiceLevelObjective>
          <puppetSLO:PuppetSLO>
            <puppetSLO:Latency>
              <value>10000</value>

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

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

            <wsag:ValueExpression> 2 </wsag:ValueExpression>
          </wsag:Penalty>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>

    </wsag:All>
  </wsag:Terms>
</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.

```
package services;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Iterator;

import density.Density;

public class WarehousePortBindingImpl {
    /*
     * The SSM object.
     */
    private info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine aMbItIoNssm;

    /*
     * The simulator is generated
     */
    private info.frantzen.testing.ssmsimulator.SSMSimulator aMbItIoNsim;

    private info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNfindSSMMessage(
            info.frantzen.testing.ssmsimulator.ssm.ServiceStateMachine ssm,
            info.frantzen.testing.ssmsimulator.ssm.MessageKind kind,
            info.frantzen.testing.ssmsimulator.ssm.Operation op)
            throws Exception {
         java.util.HashSet<info.frantzen.testing.ssmsimulator.ssm.Message> messages = ssm
               .getMessages();
         for (java.util.Iterator it = messages.iterator(); it.hasNext();) {
            info.frantzen.testing.ssmsimulator.ssm.Message m = (info.frantzen.testing.ssmsimulator.ssm.Message) it
                  .next();
            if (m.getKind() != info.frantzen.testing.ssmsimulator.ssm.MessageKind.UNOBSERVABLE) {
               if ((m.getKind() == kind) && (m.getOperation().equals(op))) {
                  return m;
               }
            }
         }
         throw new Exception(
               "Cannot␣find␣the␣input␣SSM␣message␣belonging␣to␣the␣Operation␣"
                     + op.getName() + "!");
    }

    public WarehousePortBindingImpl() throws Exception {
         /*
          * To initialise the Simulator, the following items are needed:
          */

         /*
          * The URL of the WSDL file
          */
         java.net.URL aMbItIoNWSDLUrl = new java.net.URL("Put␣here␣the␣URL␣of␣the␣Service’s␣WSDL");
         /*
          * The name of the WSDL-Service
          */
         String aMbItIoNservice = "Put␣here␣the␣name␣of␣the␣Service␣as␣in␣the␣WSDL";
         /*
          * The name of the WSDL-Port
          */
         String aMbItIoNport = "Put␣here␣the␣port␣of␣the␣Service␣as␣in␣the␣WSDL";
         /*
          * The URL of the SSM Schema Instance
          */
         java.net.URL aMbItIoNSSMUrl = new java.net.URL("Put␣here␣the␣URL␣of␣the␣SSM␣Schema␣Instance");

         /*
          * Now we can generate the SSM object. To do so, we use Zsolt’s "Minerva" library
          */
         aMbItIoNssm = hu.soft4d.jessi.ssm.SSMHandler.generateSSM(
               aMbItIoNWSDLUrl, aMbItIoNSSMUrl, aMbItIoNservice, aMbItIoNport);
         /*
          * Before we can use the SSM in the simulator, the parsers have to be attached
          * to the switches
          */
         aMbItIoNssm.attachParsersToSwitches();
         /*
          * Next we generate the socket to the treeSolver.
          */
         String aMbItIoNsolverHost = "Put␣here␣the␣URL␣of␣the␣Solver";
         int aMbItIoNsolverPort = "Put␣here␣the␣Port␣the␣Solver";
         java.net.Socket aMbItIoNsolverSocket = new java.net.Socket(
               aMbItIoNsolverHost, aMbItIoNsolverPort);
         /*
          * The treeSolver sends a welcome message, we remove it from the stream
          */
         new java.io.BufferedReader(new java.io.InputStreamReader(
               aMbItIoNsolverSocket.getInputStream())).readLine();
         /*
          * The simulator can use an external tool to display sequence diagrams
          * of the messages exchanged. // I skip this here since this takes extra
          * ressources/
          */

         /*
          * The simulator needs a logger to log to
          */
         java.util.logging.Logger aMbItIoNlogger = java.util.logging.Logger
               .getLogger("");

         /* The simulator is generated */

         aMbItIoNsim = new info.frantzen.testing.ssmsimulator.SSMSimulator(
               aMbItIoNssm, aMbItIoNsolverSocket, aMbItIoNlogger);

         /*
          * If Double variables are used we assume this models money
          * (experimental). In any case, do this:
          */
         info.frantzen.testing.ssmsimulator.types.ST_PseudoPosDouble.postPointLength = 2;
```

```java
      /*
       * Now the Simulator is ready.
       * --------------------------------------------------------------------------
       */
   }

   public services.Quote checkAvail(services.QuoteRequest r)
         throws java.rmi.RemoteException {
      long aMbItIoNinvocationTime = 0;
      services.Quote aMbItIoNreturnValue;
      try {
         aMbItIoNinvocationTime = System.currentTimeMillis();

         /*
          * Code Generated for Integration with Ambition
          */
         info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
              ssmsimulator.ssm.Operation("checkAvail", info.frantzen.testing.ssmsimulator.ssm.OperationKind.
              REQUESTRESPONSE);
         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage( aMbItIoNssm,
              info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT, aMbItIoNoperation);
         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
              ssmsimulator.ssm.Valuation();
         java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
              aMbItIoNmessage.getType();
         java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
         info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;

         /*
          * Generated Parameter 0
          */
         aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
         Object[] aMbItIoNparameterValues = new Object[2];
         aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(r.
              getProduct());
         aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_PosIntInstance(r.
              getQuantity());
         info.frantzen.testing.ssmsimulator.types.TypeInstance rInstance = new info.frantzen.testing.
              ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
         aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), rInstance);

         /*
          * The valuation is ready, we can construct an instantiated message
          */
         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
              ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);

         /*
          * This instantiated message can now be given to the simulator. Note
          * that here the simulator can potentially spot a failure, namely
          * when this message is not specified in the SSM! In that sense,
          * here we do testing.
          */
         aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);

         /*
          * Ok, the simulator knows the input. Now we need a functionally
          * correct response to this call. We first ask the simulator for all
          * currently activated output transitions.
          */
         java.util.ArrayList aMbItIoNoutputs = new java.util.ArrayList(aMbItIoNsim.getCurrentOutputSwitches());

         /*
          * 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.
          */
         boolean aMbItIoNnoSolutionFound = true;
         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNnextOutput = null;
         java.util.Random aMbItIoNrandom = new java.util.Random();
         while (!aMbItIoNoutputs.isEmpty() && aMbItIoNnoSolutionFound) {
            info.frantzen.testing.ssmsimulator.ssm.Switch aMbItIoNcandidate = (info.frantzen.testing.
                 ssmsimulator.ssm.Switch) aMbItIoNoutputs.get(aMbItIoNrandom.nextInt(aMbItIoNoutputs.size()));

            /*
             * try to find a solution, if yes, fine, if not, remove the
             * candidate
             */
            aMbItIoNnextOutput = aMbItIoNsim.findSolution(aMbItIoNcandidate);
            if (aMbItIoNnextOutput == null)
               aMbItIoNoutputs.remove(aMbItIoNcandidate);
            else
               aMbItIoNnoSolutionFound = false;
         }
         if (aMbItIoNnextOutput == null)
            throw new Exception("Failure␣in␣SSM!␣No␣output␣for␣synchronous␣input␣specified!");
         /*
          * Ok, we have now a feasible and functionally correct output:
          * nextOutput Before we send this output to the Service out there,
          * we tell so to the simulator:
          */
         aMbItIoNsim.processInstantiatedMessageNoBackup(aMbItIoNnextOutput);

         /*
          * What is left to do, is to map this instantiated message back to a
          * real returnValue.
          */
         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNreturnMessage = aMbItIoNnextOutput.getMessage()
              ;
         String aMbItIoNreturnVarName = ((info.frantzen.testing.ssmsimulator.ssm.InteractionVariable)
              aMbItIoNreturnMessage.getType().iterator().next()).getName();
         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNreturnValuation = aMbItIoNnextOutput.
              getValuation();
         info.frantzen.testing.ssmsimulator.types.TypeInstance aMbItIoNreturnInstance = aMbItIoNreturnValuation
              .getSingleInstance(aMbItIoNreturnVarName);
         String[] aMbItIoNarrayRepresentation = aMbItIoNreturnInstance.toString().split(",");
         aMbItIoNreturnValue = new services.Quote((Double.valueOf(aMbItIoNarrayRepresentation[0]).doubleValue()
              ),aMbItIoNarrayRepresentation[1],(Integer.valueOf(aMbItIoNarrayRepresentation[2]).intValue()),(
              Integer.valueOf(aMbItIoNarrayRepresentation[3]).intValue()));
         /*
          * Now send the returnValue back to the calling service. That's it.
          */
      } catch (Exception genericException) {
         throw new java.rmi.RemoteException(genericException.getMessage());
      }
      Density D = new Density();
      Double sleepValue = D
            .gaussian(25000 - puppet.ambition.Naturals
                  .asNatural(aMbItIoNinvocationTime
                        - System.currentTimeMillis()));
      try {
         Thread.sleep(sleepValue.longValue());
      } catch (InterruptedException e) {
      }
      return aMbItIoNreturnValue;
   }

   public void cancelTransact(int ref) throws java.rmi.RemoteException {
      long aMbItIoNinvocationTime = 0;
      try {
         aMbItIoNinvocationTime = System.currentTimeMillis();
         /*
          * Code Generated for Integration with Ambition
          */
         info.frantzen.testing.ssmsimulator.ssm.Operation aMbItIoNoperation = new info.frantzen.testing.
              ssmsimulator.ssm.Operation(
               "cancelTransact",
         info.frantzen.testing.ssmsimulator.ssm.OperationKind.ONEWAY);
         info.frantzen.testing.ssmsimulator.ssm.Message aMbItIoNmessage = aMbItIoNfindSSMMessage(
               aMbItIoNssm,
               info.frantzen.testing.ssmsimulator.ssm.MessageKind.INPUT,
               aMbItIoNoperation);
         info.frantzen.testing.ssmsimulator.ssm.Valuation aMbItIoNvaluation = new info.frantzen.testing.
              ssmsimulator.ssm.Valuation();
         java.util.ArrayList<info.frantzen.testing.ssmsimulator.ssm.InteractionVariable> aMbItIoNmessageType =
              aMbItIoNmessage
               .getType();
         java.util.Iterator aMbItIoNit = aMbItIoNmessageType.iterator();
         info.frantzen.testing.ssmsimulator.ssm.InteractionVariable aMbItIoNvar;

         /*
          * Generated Parameter 0
          */
         aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit
               .next();
         info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
              ssmsimulator.types.ST_PosIntInstance(
               ref);
         aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),
               refInstance);

         /*
          * The valuation is ready, we can construct an instantiated message
          */
         info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
              ssmsimulator.ssm.InstantiatedMessage(
               aMbItIoNmessage, aMbItIoNvaluation);

            /*
             * This instantiated message can now be given to the simulator. Note
             * that here the simulator can potentially spot a failure, namely
             * when this message is not specified in the SSM! In that sense,
             * here we do testing.
             */
            aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
        } catch (Exception genericException) {
            throw new java.rmi.RemoteException(genericException.getMessage());
        }
    }

    static long startWinTimeStamp = System.currentTimeMillis();

    static ArrayList<Long> faultBuffer = new ArrayList<Long>();

    static int executedFault = 0;

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

            /*
             * Generated Parameter 0
             */
            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
            info.frantzen.testing.ssmsimulator.types.TypeInstance refInstance = new info.frantzen.testing.
                 ssmsimulator.types.ST_PosIntInstance(ref);
            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(), refInstance);

            /*
             * Generated Parameter 1
             */
            aMbItIoNvar = (info.frantzen.testing.ssmsimulator.ssm.InteractionVariable) aMbItIoNit.next();
            Object[] aMbItIoNparameterValues = new Object[2];
            aMbItIoNparameterValues[0] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                 getFirstName());
            aMbItIoNparameterValues[1] = new info.frantzen.testing.ssmsimulator.types.ST_StringInstance(adr.
                 getLastName());
            info.frantzen.testing.ssmsimulator.types.TypeInstance adrInstance = new info.frantzen.testing.
                 ssmsimulator.types.ComplexTypeInstance(aMbItIoNparameterValues);
            aMbItIoNvaluation.addSingleValuation(aMbItIoNvar.getName(),adrInstance);

            /*
             * The valuation is ready, we can construct an instantiated message
             */
            info.frantzen.testing.ssmsimulator.ssm.InstantiatedMessage aMbItIoNim = new info.frantzen.testing.
                 ssmsimulator.ssm.InstantiatedMessage( aMbItIoNmessage, aMbItIoNvaluation);

            /*
             * This instantiated message can now be given to the simulator. Note
             * that here the simulator can potentially spot a failure, namely
             * when this message is not specified in the SSM! In that sense,
             * here we do testing.
             */
            aMbItIoNsim.processInstantiatedMessage(aMbItIoNim);
        } catch (Exception genericException) {
            throw new java.rmi.RemoteException(genericException.getMessage());
        }
        long winSize = 120000;
        int maxFault = 3;
        long currentTimeStamp = System.currentTimeMillis();
        for (int i=0; i<faultBuffer.size();i++){
            if (currentTimeStamp - faultBuffer.get(i) >= winSize){
                faultBuffer.remove(i);
            }
        }
        if (faultBuffer.size() < maxFault){
            Density d = new Density();
            double dv = d.gaussian(100);
            if (dv > 50) {
                String fCode = "Server.NoService";
                String fString = "PUPPET:␣No␣target␣service␣to␣invoke!";
                org.apache.axis.AxisFault fault = new org.apache.axis.AxisFault(
                      fCode, fString, "", null);
                aMbItIoNsim.undo();
                faultBuffer.add(currentTimeStamp);
                throw fault;
            }
        }
     }
  }
```

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

[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.