Verification tools for service descriptions

Project acronym: COMPAS
Project name: Compliance-driven Models, Languages, and Architectures for Services
Call and Contract: FP7-ICT-2007-1
Grant agreement no.: 215175
Project Duration: 01.02.2008 – 28.02.2011 (36 months)

Co-ordinator: TUV Technische Universitaet Wien (AT)
Partners: CWI Stichting Centrum voor Wiskunde en Informatica (NL)
           UCBL Université Claude Bernard Lyon 1 (FR)
           USTUTT Universitaet Stuttgart (DE)
           TILBURG UNIVERSITY Stichting Katholieke Universiteit Brabant (NL)
           UNITN Universita degli Studi di Trento (IT)
           TARC-PL Telcordia Poland (PL)
           THALES Thales Services SAS (FR)
           PWC Pricewaterhousecoopers Accountants N.V. (NL)

This project is supported by funding from the Information Society Technologies Programme under the 7th Research Framework Programme of the European Union.
Project no. 215175

COMPAS

Compliance-driven Models, Languages, and Architectures for Services

Specific Targeted Research Project
Information Society Technologies

Start date of project: 2008-02-01  Duration: 36 months

D3.3 Verification tools for service descriptions

Revision 1.0

Due date of deliverable: 2009-12-31
Actual submission date: 2009-12-22

Organisation name of lead partner for this deliverable:
UCBL, University of Claude Bernard Lyon1, LIRIS (FR)

Contributing partner(s):
CWI, Stichting Centrum voor Wiskunde en Informatica (NL)
TUV Technische Universität Wien, AT
USTUTT Universitaet Stuttgart, DE

| Project funded by the European Commission within the Seventh Framework Programme |
| Dissemination Level                                                                 |
| PU  | Public                             | X |
| PP  | Restricted to other programme participants (including the Commission Services) |
| RE  | Restricted to a group specified by the consortium (including the Commission Services) |
| CO  | Confidential, only for members of the consortium (including the Commission Services) |
History chart

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Changed page(s)</th>
<th>Cause of change</th>
<th>Implemented by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2009-12-11</td>
<td>All sections</td>
<td>New document</td>
<td>UCBL</td>
</tr>
<tr>
<td>0.2</td>
<td>2009-12-12</td>
<td>3.7</td>
<td>New section</td>
<td>UCBL</td>
</tr>
<tr>
<td>0.3</td>
<td>2009-12-14</td>
<td>All sections</td>
<td>Various minor revisions</td>
<td>UCBL</td>
</tr>
<tr>
<td>0.4</td>
<td>2009-12-17</td>
<td>All sections</td>
<td>Alignment to COMPAS format</td>
<td>UCBL</td>
</tr>
<tr>
<td>0.5</td>
<td>2009-12-18</td>
<td>All sections</td>
<td>Various minor revisions</td>
<td>UCBL + partners</td>
</tr>
<tr>
<td>1.0</td>
<td>2009-12-22</td>
<td></td>
<td>Approve &amp; Release</td>
<td>TUV</td>
</tr>
</tbody>
</table>

Authorisation

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Company/Name</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prepared</td>
<td>UCBL</td>
<td>2009-12-18</td>
</tr>
<tr>
<td>2</td>
<td>Approved</td>
<td>TUV</td>
<td>2009-12-22</td>
</tr>
<tr>
<td>3</td>
<td>Released</td>
<td>TUV</td>
<td>2009-12-22</td>
</tr>
</tbody>
</table>

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

All rights reserved.

The document is proprietary of the COMPAS consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights.

This document reflects only the authors’ view. The European Community is not liable for any use that may be made of the information contained herein.
Contents

List of figures .................................................................................................................................. 4
List of listings .................................................................................................................................. 5
Abstract ............................................................................................................................................ 6

1. Introduction ................................................................................................................................. 6
   1.1. Purpose and scope................................................................................................................ 6
   1.2. Document overview ............................................................................................................. 6
   1.3. Definitions and glossary ...................................................................................................... 6
   1.4. Abbreviations and acronyms ............................................................................................... 6

2. Verification tools into the COMPAS architecture ........................................................................ 6

3. Business protocol editor ............................................................................................................. 7
   3.1. Installation ............................................................................................................................ 7
       (a) Software environment .................................................................................................. 7
       (b) Installing and editing ................................................................................................... 7
       (c) Software libraries ......................................................................................................... 8
   3.2. Construction of WSDL file ................................................................................................. 8
   3.3. Definition of credential and policy constraints .................................................................. 8
   3.4. Checking compatibility ....................................................................................................... 9
   3.5. Checking the replaceability ................................................................................................. 9
   3.6. Wizards ............................................................................................................................... 10
   3.7. Compatibility and replaceability checking library .......................................................... 10
       3.7.1. Constraint automata integration ................................................................................ 10
       3.7.2. Ant tasks ...................................................................................................................... 10
       3.7.3. Example Ant build script ........................................................................................... 11

4. Scenarios .................................................................................................................................... 11
   4.1. Conversion Algorithm ...................................................................................................... 12
   4.2. Product Automata and Compatibility checking ............................................................... 14
   4.3. Intersection Automata and Replaceability checking ....................................................... 20
   4.4. Checking of messages exchanged ................................................................................... 23

5. Reference documents ................................................................................................................ 25
   5.1. Internal documents ............................................................................................................ 25
   5.2. External documents ........................................................................................................... 25

List of figures

Figure 1 Part of the overall COMPAS architecture ........................................................................ 7
Figure 2  Business Protocol p1 with implicit and explicit transitions. .................................. 12
Figure 3  Wizard of new web service business converted protocol file. .............................. 13
Figure 4  Web Business Protocol Converted p1. .................................................................. 13
Figure 5  Graphical representation of ACP ontology .............................................................. 14
Figure 6  A sample web business protocol named b1ForProduct.wsprotocol. .................... 16
Figure 7  A sample web business protocol named b2ForProduct.wsprotocol. .................... 17
Figure 8  Wizard of new web service business product protocol file. .................................. 18
Figure 9  Web service business product protocol derived from b1ForProduct.wsprotocol and b2ForProduct.wsprotocol. .............................................................. 18
Figure 10  Compatibility checking for b1ForProduct.wsprotocol and b2ForProduct.wsprotocol. ............................................................................................................ 19
Figure 11  Web service business protocol named b1ForIntersection.wsprotocol. .............. 20
Figure 12  Web service business protocol named b2ForIntersection.wsprotocol. .............. 21
Figure 13  Wizard of new web service business intersection protocol file. ......................... 22
Figure 14  Web service business intersection protocol derived from b1ForIntersection.wsprotocol and b2ForIntersection.wsprotocol. .............................................................. 22
Figure 15  Replaceability checking for b1ForProduct.wsprotocol and b2ForProduct.wsprotocol. ............................................................................................................ 23
Figure 16  Web service business protocol named p1Message.wsprotocol. ......................... 24
Figure 17  Web service business protocol named p2Message.wsprotocol. ......................... 24

List of listings

Listing 1  Property file ................................................................................................................ 11
Listing 2  Representation of ontology.owl file. ......................................................................... 16
Abstract

This deliverable presents the design of a software package which includes an Eclipse plugin for editing business protocols and a library for checking compatibility and replaceability of business protocols and constraint automata, which can handle ontology access control specification as an example of data constraints.

1. Introduction

1.1. Purpose and scope

The major goal of deliverable D3.3 is developing process verification tools. These tools include the checking of compatibility and replaceability between two services. In particular, they can be used during service selection and composition, typically at deployment time.

1.2. Document overview

The remainder of this document is structured as follows: Section 2 describes the role of the verification tools on the COMPAS architecture by highlighting its interaction with the other components of the COMPAS Architecture defined in [DA.1]. Section 3 describes the business protocol editor tool by describing the components of the software and their use. Section 4 gives set of scenarios for exploring the functionality of the verification tools.

1.3. Definitions and glossary

The most important terminology concerning the COMPAS project is listed on the public COMPAS Web-Site [D7.1] available at http://www.compas-ict.eu section terminology. This helps to make the overall COMPAS approach more comprehensive for the general public.

1.4. Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>ACP</td>
<td>Access Control Policy</td>
</tr>
<tr>
<td>DL</td>
<td>Description Language</td>
</tr>
<tr>
<td>SAWSDL</td>
<td>Semantic annotation Web Service Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>

2. Verification tools into the COMPAS architecture

This section describes the verification tools context with regard to COMPAS Architecture [DA.1]. It clarifies the integration between verification tools and other components of the architecture. COMPAS Architecture provides relevant prototypes, tools, and technologies, with the use cases provided by industry partners in order to demonstrate the pragmatic use of COMPAS contributions.
Figure 1 presents process verification tools and its interaction with other components. WP3 is responsible for the development of three main components of the architecture: process verification tools, Reo editor and converters. The Reo editor edits process models which is presented by the compliance language (WP2) or from process repository (WP4) and convert it to Reo, the process verification tools apply set of checking operations as like compatibility and replaceability checking, and the converters use for converting process to Reo model.

![Figure 1 Part of the overall COMPAS architecture](image)

3. Business protocol editor

3.1. Installation

(a) Software environment

- Eclipse (SDK, Ganymede) plugged with the Graphical editing framework draw2D and Graphical Editing Framework GEF tools.
- JRE 1.6

(b) Installing and editing

1. Update the eclipse with the business protocol editor by pointing to the folder containing this plugin (Plugin)
2. Open the eclipse in the business protocol editing perspective.
3. Import one of the businessprotocolTest project from “Compas 1.1.0 definitive/eclipse project” which contains the compatibility and replaceability classes.

4. Use the editor to design two protocols.

5. Adjust the run configuration for the project by passing the two business protocol files and the ontology file(owl file) as an input arguments

6. Run with this configuration.

7. The result of the checking will be appearing in the output console.

(c) Software libraries

A business protocols model library: contains an object-oriented model for representing and manipulating business protocols. While abstracting from the physical representation of protocols (e.g., files, databases, ...) , it contains a simple XML persistence class that allows protocols to be stored and read from various XML. Temporal constraints are explicitly supported by the library. They are stored as plain strings in the model that are attached to the transitions of a protocol. It is used also a parser written using AntLR [1] for obtaining the object model that corresponds to a constraint in a plain string and vice-versa.

A protocol manipulation library: implements the two main checking algorithms, compatibility and replaceability between business protocols using access control policy.

A set of Eclipse plug-ins: which provides a business protocol editor based on a set of customized SWT/JFace figures and the Eclipse Graphical Editing Framework [5]. This can be useful in order to interface with the business protocol model editor. The analysis components provide a visual interface for either invoking the manipulation and comparison operators.

3.2. Construction of WSDL file

Web Service Description Language file (WSDL) [CMR+07] file is important in parameter in a real checking scenarios because it presents the schema elements that defines a messages over an operation on the business protocol. In our work, WSDL file is constructed automatically during the compatibility and replaceability checking in order to check the compatibility between schema elements of the messages exchanged between two different timed business protocols. In this way, based on a predefined timed business protocol, we defined some classes that perform this construction.

A schema element can have a semantic annotation tag. Semantic Annotation are used in order to express the fact that an element in a WSDL file is belonging to a predefined ontology. So semantic annotation defines a mapping with ontology and in this implementation is utilized for mapping credential constraint element (or policy constraint element) inside the schema element of a message provided by the protocol. In this way it is possible to understand the compatibility and replaceability checking also in terms of ACP. The ontology is represented in a file, with extension *owl, and represents a taxonomy with individuals that can be defined with any ontology tool. In this implementation, we used the Protégé as external tool to define ontology taxonomy with individuals. In this sense, the WSDL file becomes a SAWSDL file, a WSDL file with Semantic Annotations [SA07]. The SAWSDL file is useful in the compatibility and replaceability checking.

3.3. Definition of credential and policy constraints

Development of suitable access control models, able to restrict access to Web services to authorized users is an important issue. As depicted in the paper, Access Control Policy (ACP)
languages grounded in Semantic Web technologies allow policies to be described over heterogeneous domain data and promote common understanding among participants who might not use the same information model. There are two main advantages of using ontology in policy specification and management:

- Ease policy specification and management by sharing policies for common attributes, composing and overriding policies.
- Protect sensitive information by avoiding information leaking request and answering unnecessary request.

For that reason, Description Logics (DLs) [NO05] as policy formalisms technique can be used to present access control policy ontology, so that the access control policies are described as concepts describe set of credentials as individuals. Considering ACP of web services as ontology will enables to use ontology alignment tools to find classes of data that are “semantically equivalent”. We extended the timed constraints and defined credentials or policies constraints.

### 3.4. Checking compatibility

We construct the product automata derived from two timed business protocol and check the compatibility in terms of messages exchanged using it; verify the co-accessibility of states in the product automata; and finally checking the compatibility in terms of ACP using the created product automata. In this latter part, the implementation also constructs the credential cumulative paths and then checks the subsumption between credential constraints, even if there are present in the product automata. In this sense, the role of the reasoner is important because it used SAWSDL file defined for every business protocol in order to extract the information relative to each constraints, identified in the schema element of a message, and then verify the subsumption. In this way the compatibility check is establish between two different timed business protocol and it answers true if the two protocol tested are compatible and false otherwise.

### 3.5. Checking the replaceability

We construct the intersection automata derived from two timed business protocol and check the replaceability in terms of messages using it; verify the co-accessibility of states in the intersection automata; and finally checking the replaceability in terms of ACP using the created intersection automata. However the cumulative paths defined are different from the case of the compatibility, it calculates the policy cumulative paths from the first protocol and the credential cumulative paths from the second protocol. This is done because in this case it has to check the replaceability of the first protocol with the second protocol and so policy constraints are important not only for a complete cumulative check presence of each policies of the first protocol inside the second one, but also in order to do not replace the first protocol with a second protocol that introduce more restrictions on policies in terms of a possible interaction with other external protocols. Even in this case the role of the Reasoner is important in order to check the subsumption not only for credentials but also for policies.
3.6. Wizards

- Converted Business Protocol: which represents an Eclipse Wizard that permits to apply the conversion algorithm to a business protocol given in input and so to generate the Explicitly Timed Business Protocol.
- Product Automata Business Protocol: which is an Eclipse Wizard that permits to construct a Product Automata from two Business Protocol given in input. It also constructs the relative SAWSDL file as a result of the relative ontology given as input.
- Intersection Automata Business Protocol: which is an Eclipse Wizard that permits to construct an Intersection Automata from two Business Protocol given in input. Even in this case it also constructs the relative SAWSDL file as a result of the relative ontology given as input.

3.7. Compatibility and replaceability checking library

The directory fr.cnrs.liris.wsprotocol.analysis is an eclipse project containing the library code source and binaries together with examples.

The library is provided as a jar file wsprotocolanalysis.jar located in the lib subdirectory. To run, it requires JRE 1.6 and the various libraries including the Protégé / Pellet libraries for ontology reasoning. All the required libraries are included in the lib subdirectory.

3.7.1. Constraint automata integration

The XSL stylesheet ca2bp.xsl allows converting from constraint automata format to explicitly timed business protocol format. The convertible constraint automata class is currently limited as follows: (1) it should use at least two ports, one named i, is used for input messages while the other(s) are considered as carrying output messages, the constraint on ports in transitions expressing the message name; (2) the automata should use only one clock which should be reset on each transitions, which corresponds to the time constraints expressivity of the current explicitly timed business protocol model.

3.7.2. Ant tasks

The library includes two Ant task fr.cnrs.liris.wsprotocol.ant.CompatibilityTask for checking compatibility and fr.cnrs.liris.wsprotocol.ant.ReplaceabilityTask for checking replaceability. The two tasks can be parameterized as follows:

- Attributes protocol1 and protocol2 specifies the protocol files to compare. The replaceability task check whether protocol1 can replace protocol2.
- Attribute ontology specifies the OWL file containing the ontology description.
A convenience property file, wsprotocoltasks.properties, is provided to ease the definition of those tasks in the ant build file. Using this property file, the tasks can be defined using:

```xml
<target name="taskdefs">
  <property file="wsprotocoltasks.properties" />
  <taskdef name="compatibilitytask"
    classname="${wsprotocolstasks.compatibilityclass}" classpath="${wsprotocolstasks.classpath}" />
  <taskdef name="replaceabilitytask"
    classname="${wsprotocolstasks.replaceabilityclass}" classpath="${wsprotocolstasks.classpath}" />
</target>
```

**Listing 1  Property file**

### 3.7.3. Example Ant build script

An example of ant build file, example.build.xml, is provided to show how tasks can be used. In this example, the Ant target compatibilityCheck checks the compatibility of two business protocols, examples/b1forProduct.wsprotocol and examples/b2forProduct.wsprotocol using the examples/prova.owl ontology. The target replaceabilityCheck checks if the constraint automata examples/caProtocol.ea can replace the protocol examples/caProtocolB.ca. This target depends on the CA2BP task that converts the constraint automata to business protocols using the XSL style sheet. Finally the target replaceabilityCheckFail shows the failure of a replaceability check by inverting the two protocols tested in replaceabilityCheck.

### 4. Scenarios

This section presents some business protocols scenarios related to the work carried out, in particular for the conversion algorithm, the compatibility and replaceability checking.

Figure 2 presents a business protocol with implicit and explicit transitions scenario.

![Business Protocol p1 with implicit and explicit transitions.

In Figure 1 there are several states in which s1 is the initial state and s6 is the final state. The properties of each item are shown in the Properties Tab of Eclipse in the business protocol editing perspective. For example, in figure 1 time constraints for transition \( T_0 \) is \( C-Invoke(k<8) \) which means that this transition have a time window from zero to eight to triggers since the service in the state S1.

Our plug-in contains the conversion wizard, therefore it is possible to create a Web Service Business Converted protocol file (business protocols contains only explicit transitions). This is exposed in Figure 3.
In this sense, the wizard produces a new file named `p1Converted.wsprotocol` which is showed in Figure 4.

![Wizard of new web service business converted protocol file.](image)

**Figure 3** Wizard of new web service business converted protocol file.

![Web Business Protocol Converted p1.](image)

**Figure 4** Web Business Protocol Converted p1.
Time constraints over transitions will be in the following forms (I stands for Infinity).
- $T_0 : [0, 2[$
- $T_3 : [0, 2[$
- $T_{3\_Conv3} : [4, 6[$
- $T_{3\_Conv4} : [6, I[$
- $T_{3\_Conv1} : [1, 3[$
- $T_{3\_Conv2} : [3, I[$
- $T_5\_Conv5 : [2, I[$
- $T_5 : [0, I[$

### 4.2. Product Automata and Compatibility checking.

In this section, the plugin wizard for the construction of the product automata and the relative compatibility checking is explained. Before starting it is necessary to expose the ontology file `ontology.owl` used for that scenario. The ontology taxonomy is described in the Figure 5, while the code part shows the individuals used for the compatibility and replaceability example.

![Graphical representation of ACP ontology](image)

**Figure 5** Graphical representation of ACP ontology

```xml
<?xml version="1.0"?>
<rdf:RDF
<protege:ontology
```

File: D3.3_Verification-tools-for-service-descriptions.docx Page 14 of 25
Listing 2  Representation of ontology.owl file.

As it is possible to see in the last part of the Listing 2, there are some individuals that belong to the class taxonomy. These are useful for future examples especially in the compatibility and replaceability checking.

In Figure 5 is presented a simple web business protocol useful for compatibility checking.

![Figure 5](image)

Figure 6  A sample web business protocol named b1forProduct.wsprotocol.

Where each transitions are explained in the following (I stands for Infinity):
- o T0 : Credential Constraint= P-C(P1), Time Constraint= T-Interval([0,5]);
- o T2 : Credential Constraint= P-C(L1);
- o T3 : Time Constraint= T-Interval([0,3]);
- o T6 : Credential Constraint= P-C(Sc1), Time Constraint= T-Interval([0,1]).
A second business protocol is showed in the Figure 7.

![Figure 7](image)

**Figure 7** A sample web business protocol named b2forProduct.wsprotocol.

Where each transitions are explained in the following (I stands for Infinity):

- o T8 : Time Constraint= T-Interval([0,6[);
- o T10 : Policy Constraint= P-C((L1 && P1));
- o T11 : Credential Constraint= P-C(J1), Time Constraint= T-Interval([0,3[);
- o T12 : Credential Constraint= P-C(C1);
- o T14 : Time Constraint= T-Interval([0,I[).

The Web Service Business Product protocol file wizard, exposed in Figure 7, creates the Product Automata illustrated in Figure 8.
Figure 8  Wizard of new web service business product protocol file.

Figure 9  Web service business product protocol derived from b1ForProduct.wsprotocol and b2ForProduct.wsprotocol.
Where each transitions are explained in the following (I stands for Infinity):

- T24 : Credential Constraint= P-C(P1), Time Constraint= T-Interval([0,5[);
- T26 : Credential Constraint= P-C(L1), Policy Constraint= P-(L1 & P1);
- T27 : Credential Constraint= P-C(J1), Time Constraint= T-Interval([0,3[);
- T28 : Credential Constraint= P-C(C1);
- T29 : Credential Constraint= P-C(Sc1), Time Constraint= T-Interval([0,1[).

As a result of the construction, it is also constructed the SAWSDL file relative to the two business protocol involved. The output exposed by the compatibility checking is showed in the Figure 10.

![Figure 10 Compatibility checking for b1ForProduct.wsprotocol and b2ForProduct.wsprotocol.](image)

This simple example recognizes also the utility of cumulative credential path determinate for the protocol b1ForProduct.wsprotocol, as can be seen from the transitions description. It is obvious that if some time constraints, credential constraints, policy constraints are changed and not satisfy the compatibility checking, the algorithm will capture it.
4.3. Intersection Automata and Replaceability checking.

Even in this case it can be done the same things, obviously with web business protocol appropriated. In the Figure 11 shows a web service business protocols useful for a replaceability checking.

![Figure 11 Web service business protocol named b1ForIntersection.wsprotocol.]

Where each transitions are explained in the following (I stands for Infinity):
- T2 : Credential Constraint P-C(L1), Time Constraint = T-Interval([0,5]);
- T3 : Policy Constraint = P-C(Sc1), Time Constraint = T-Interval([0,3]);
- T4 : Policy Constraint = P-C(P1);
- T12 : Credential Constraint = P-C(C1);
- T14 : Time Constraint = T-Interval([0,1]).

A second business protocol is showed in the Figure 12.
Figure 12 Web service business protocol named b2ForIntersection.wsprotocol.

Where each transitions are explained in the following:

- T6 : Credential Constraint P-C(L1), Time Constraint= T-Interval([0,5]);
- T8 : Time Constraint= T-Interval([0,5]);
- T9 : Policy Constraint= P-C(Sc1), Time Constraint= T-Interval([0,2]);
- T10 : Credential Constraint= P-C(P1);
- T11 : Credential Constraint= P-C(L1).

The Web Service Business Intersection protocol file wizard, exposed in Figure 13, creates the Intersection Automata illustrated in Figure 14.
Figure 13 Wizard of new web service business intersection protocol file.

Figure 14 Web service business intersection protocol derived from b1ForIntersection.wsprotocol and b2ForIntersection.wsprotocol.
Where each transitions are explained in the following:

- T48: Credential Constraint $P-C(L1)$;
- T50: Credential Constraint= $P-C(L1)$, Time Constraint= $T-Interval([0,5])$;
- T51: Policy Constraint= $P-C(Sc1)$, Time Constraint= $T-Interval([0,2])$;
- T52: Policy Constraint= $P-C(P1)$;
- T53: Credential Constraint= $P-C(L1)$.

As done for compatibility, it is also constructed the SAWSDL file relative to the two business protocol involved. The output exposed by the compatibility checking is showed in the Figure 15.

![Figure 15 Replaceability checking for b1ForProduct.wsprotocol and b2ForProduct.wsprotocol.](image)

The output is right with the definitions of the two web business protocol, in fact they are not replaceable for time interval reason, because transition T3 from the first protocol has a time constraint $T-Interval([0,3])$, while the transition T9 from the second protocol has a time constraint $T-Interval([0,2])$, so T1 has a time constraint that is not contained or equal to T2 and not satisfy the replaceability checking. It is obvious that if this time constraint is changed in order to satisfy the time containment test, then the checking returns that they are replaceable, except that other algorithm parts are satisfied.

4.4. Checking of messages exchanged.

As a further example, here is presented a scenario in which two business protocol are not compatible in terms of messages exchanged. The same discussion can be done also in a replaceability checking scenario. The two protocols are showed in Figure 16 and Figure 17.
As expected the output of a compatibility checking shows that the two protocols involved are not compatible. The same case can be applied for a replaceability checking.
5. Reference documents

5.1. Internal documents

[DA.1] COMPAS Architectural Walkthroughs and Evaluation Metrics


5.2. External documents


