Design and Verification of Web Services Compositions *

Enrique Martínez, María Emilia Cambronero, Gregorio Díaz, and Valentín Valero
Escuela Politécnica Superior de Albacete. Universidad de Castilla-La Mancha
Departamento de Sistemas Informáticos
Campus Universitario s/n. 02071. Albacete, SPAIN
{MEmilia.Cambronero,Gregorio.Diaz,Valentin.Valero}@uclm.es
{emartinez}@dsi.uclm.es

Abstract

In this work we explain how to perform the design and verification of a Web Services composition with time restrictions using the Web Services Translation tool, WST for short. We pay special attention to Web Services systems with timing restrictions. As illustration, we use a case study called “Form Generator Service”, a system that allows developers to manage a form repository by means of several Web Services. We show how to design this system and the verification of some properties on it, using a Timed Automata representation.

1. Introduction

The use of the Internet for doing business is becoming more and more important in the last years. This is because Web technologies allow providers to offer services in a cheaper, faster and more dynamic way. Web Services are a collection of functions that are packaged together and published in the network in a way they can be used by other programs. This is possible by the use of multiple standardized protocols in which Web Services rely on (Figure 1). This is an emerging technology, so there is still a lack of techniques and tools supporting the composition of Web Services.

Two main languages are being used in the Web Services composition framework: WS-BPEL (Web Services Business Process Execution Language [2]) at the orchestration level, and WS-CDL (Web Services Choreography Description Language [1]), at the choreography level. In brief the Orchestration describes the particualr behaviour of each participant in a Web Service and the Choreography describes the global behaviour.

* Supported by the Spanish government (cofinanced by FEDER founds) with the project TIN2006-15578-C02-02, and the JCCLM regional project PAC06-0008-6995. The first author is supported by the European Social Fund and the Junta de Comunidades de Castilla-La Mancha (Spain).

Figure 1. Stack of protocols for Web Services

One important aspect that must be covered when describing and analysing Web Services compositions is that of timing restrictions. These are quantitative restrictions that Web Services must fulfill in a specific scenario, which are essential for the correct operation of Web Services compositions. Systems with timing restrictions are the core of governmental, financial and industrial systems. The response time determines the grade of efficiency, correctness, and user satisfaction. Thus, in our tool, WST (Web Services Translation), timing aspects are considered as one of the main issues to be described and analysed. The aim of this tool is to deal with the design, implementation, verification and validation of Web Services compositions with time restrictions. In the design phase the Unified Modeling Language (UML 2.0 [3]) is used to model the system through a sequence diagram editor. The Web Services Choreography Description Language (WS-CDL) is used to obtain a choreography that corresponds to the provided sequence diagram. For the verification and validation phase we use a Timed Automata representation of the system, which is automatically obtained from the WS-CDL description, and the model checker of the UPPAAL tool [4] is used for validation and verification purposes.

Thus, using the UPPAAL tool jointly with the WST tool, we can detect some mistakes in the design phase, which can be fixed before starting the implementation. This technique reduces the cost of the final system since we are able to
detect and fix many bugs at the design phase.

The outline of the paper is as follows. Section 2 is about related work. A short explanation of how to use the WST tool is shown in Section 3. Section 4 presents the “Form Generator Service” case study, explaining how to design and verify the system. Finally, in Section 5 we show the conclusions and our future work.

2 Related Work

In [13] the problem of choreography correctness is addressed through the use of TIOTS (Timed Input Output Transition Systems) corresponding to WS-BPEL specifications. An algorithm that checks the compatibility between two TIOTS is proposed. The choreography is considered as correct if all the partners are compatible.

In [14] a conceptual model for representing Web Services with time restrictions based on FSA’s (Finite State Automata) is presented. A language called WSTL (Web Service Transition Language) that relies on this model is also introduced.

A tool called WSAT is presented in [7]. This tool is used for analyzing and verifying specifications of Web Services compositions through model-checking techniques. The language used in this work is WS-BPEL. The WS-BPEL specifications are translated into Guarded Finite State Automata (GFSA) that can be opened by the SPIN model-checker [9].

In [8] another tool called EA4B is presented. This tool generates an execution log for WS-BPEL. This logs can be used for monitoring or debugging purposes. EA4B can also be integrated with the WSAT tool.

Another tool called WS-VERIFY is shown in [10]. This tool also focuses on the analysis of WS-BPEL. In this case, the specification is translated into Web Services Timed State Transition Systems (WSTTS), which are similar to Timed Automata. The model-checker used in this case is NuSMV [11].

Finally, the design and implementation of a tool called WS-Engineer is presented in [12]. This tool is used for the verification of the implementation of Web Services compositions. It works by verifying if some specified properties are fulfilled by the composition.

3 Web Service Translation tool (WST)

The WST tool (http://www.dsi.uclm.es/retics/WST/) is a framework that allows developers to model and verify systems with time restrictions. The methodology that WST uses works as follows:

- First, the system is modelled by means of UML 2.0 sequence diagrams, by using an editor.

- Afterwards, these diagrams are converted into WS-CDL choreography specifications automatically. The UML 2.0 sequence diagrams are translated into WS-CDL by using the XML Metadata Interchange (XMI) exporting capability of WST. Thus, once the sequence diagram has been exported as an XMI document, we can use the Transform button in the second tab. By means of this button a number of XSL (Extensible Stylesheet Language [5]) rules are used in cascade to obtain the WS-CDL choreography description. These rules transform each section of the XMI document into a different element of the WS-CDL document. For instance, an object in the UML 2.0 sequence diagram is translated into the following WS-CDL entity:

```
<roleType name="name">
  <behavior name = name interface="iname" /></>
</roleType>
```

- The resulting WS-CDL specifications are translated into UPPAAL Timed Automata. This translation can be performed by using the Transform button in the third tab. Again, a number of XSL stylesheets are used in cascade in order to obtain the Timed Automata representation supported by the UPPAAL tool. These rules translate each WS-CDL element into an XML element of the Timed Automata representation. The formalities of this translation are rather complex, and they are explained in [6]. As illustration, a single interaction of WS-CDL, with the syntax:

```
<interaction name="iname" ...>
  <participate relationshipType="QName ..."/>
  <exchange name="NCname" ...>
    <send variable="XPath-expression" ... />?
    <receive variable="XPath-expression" ... />
  </exchange>>
  <timeout time-to-complete="..."/>
</interaction>
```

is translated into Timed Automata as follows:
Finally, the UPPAAL simulation engine and the model checker are used to validate and verify the system behaviour. If we detect any problem in this step, we return to the first step to correct the model.

The WST tool interface is divided into three different tabs to cover the full functionality of the tool:

1. The RT-UML DIAGRAM tab allows users to model the sequence diagrams that correspond to their systems. These diagrams can be saved, opened, deleted or exported to an XMI document for translation.

2. The RT-UML2WS-CDL tab allows users to convert the XMI representations of the diagrams into WS-CDL specifications. The resulting choreographies can be saved for future use.

3. The WS-CDL2TimedAutomata tab allows users to translate WS-CDL choreographies into XML documents describing Timed Automata. These final documents can then be saved, and immediately used in the UPPAAL tool.

4 Case Study: A Form Generator Service

Web Services are being developed for multiple purposes, to create blogs, to do searches, to purchase goods, to process data, to manage information or to create new Web contents. In the last category we can include the Form Generator Service. The aim of this service is to allow developers to manage a form repository. The service provides interfaces for creating new forms, deleting old forms, modifying existing forms, and recovering previously created forms. Developers can thus easily reuse a form when they are developing a Web site.

In this case study we focus our attention on the function of creating new forms in the repository. Figure 2 shows the sequence diagram that we have created for this function with the WST UML-editor. In this diagram we can distinguish three different parties interacting:

- The User is the developer who wants to create a new form in the repository. He has to send the name of this new form to the FormService and, after that, all the information about the fields that compose the form.
The **FormService** is the Web Service that manages the creation of the new form. It plays the role of the intermediary between the User and the DatabaseService, making the requests to the DatabaseService and sending the responses to the User.

The **DatabaseService** is the service that controls the access and modifications over the database where the information about the forms is stored. It guarantees the consistency and integrity of the database.

In the process of creating a new form we can distinguish two different phases:

1. The User has to choose a name for his new form. A request with this name is made to the database. If the name already exists a new name has to be chosen. Otherwise, the name of the form is stored in the database.

2. The User has to send all the information about the fields that compose the new form. If the DatabaseService spends more than 2 minutes before sending a response to the FormService a timeout error is sent to the User. Otherwise, an acknowledgement of the form creation is sent to the User.

Once the system has been modelled as a sequence diagram, it is exported to an XMI document. After that, the model is transformed into a WS-CDL specification by pressing the Transform button on the RT-UML2WS-CDL tab of the tool. Figure 3 depicts the XMI for this case study and the corresponding WS-CDL document.

After obtaining the WS-CDL document, we can move to the WS-CDL2TimedAutomata tab to perform the final transformation into Timed Automata. Figure 4 shows this conversion, from the WS-CDL specification to the resulting XML document with the Timed Automata specification.

These Timed Automata are synchronized through the use of channels for the message interchanges in the original diagram. Guards are used to model time restrictions (e.g. waiting time less than 2 minutes) and to control external choices (e.g. if the form name already exists). Variable updates in edges are used to reset clocks (e.g. initialization of waiting time) and to fix new values of integer and boolean variables.

When we open the resulting Timed Automata in the UPPAAL tool, we can perform a simulation of some possible behaviours of the system or we can verify some properties with the aim of proving that we have a sound system.
In this case, the properties that we intend to verify are the following:

1. We want to see if it is possible for the User party to reach the state where the form creation has finished successfully. The formula to check this property in UPPAAL is the first one in Figure 8. After pressing the checking button, we obtain that this formula is satisfied.

2. We want to see if when the FormService party reaches the state corresponding a timeout error, the value of the clock controlling to the waiting time is really greater than or equal to 2 minutes. The formula to check this property is the second one in Figure 8. In this case, the verifier shows again that the formula is satisfied.

3. We want to prove that when the User party reaches the state that begins the creation of the form fields, the value of the boolean variable controlling if the name of the form already exists is false. The formula to check this property is the third one in Figure 8. Again, the verifier shows that this formula is satisfied.

4. We want to see if after receiving a timeout error, the User party is asked to send the form information again. The formula for this property is the fourth one in Figure 8. In this case, the verifier shows that the formula is not satisfied. At this point, we should go back to our design and modify it. After that, we would repeat the translation process and we would try to verify the properties again.

5 Conclusions and future work

In this paper we have presented the WST tool and a case study to illustrate how this tool can be used in order to design and verify a Web Services system with time restrictions. The design phase is covered by using UML 2.0 sequence diagrams, and these diagrams are automatically translated into WS-CDL representations, which are in turn translated into Timed Automata, in a format supported by the UPPAAL tool.

Then, we consider that WST can be of interest for the software engineers in the process of Web Services design, with the purpose of validation and verification of the system design before starting its implementation.

As future work, we are planning to extend the functionality of WST with some new capabilities, for instance, we are working in a translation from WS-CDL to a timed model of Petri Nets (similar to Merlin’s Nets), for which a more compact representation can be obtained, in comparison with the Timed Automata representation, and for which there are a number of tools that support the model and that can be used for simulation and verification purposes.

References


