WST: A Tool for the Translation of WS-CDL into Timed Automata *

María-Emilia Cambronero, Valentín Valero, Inmaculada Aparicio, Gregorio Díaz, and Juan-José Pardo

Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha
Escuela Politécnica Superior de Albacete. 02071 - SPAIN
[emicp,valentin,inmaculada,gregorio,jpardo]@info-ab.uclm.es

Abstract. In this paper, we introduce a tool, WST (Web Services Transformation tool), for the translation of WS-CDL documents into Timed Automata. The purpose of this translation is to obtain Web Services description models with Timed Automata for validating and verificating Web Services with time restrictions. For that, we use UPPAAL tool, which is used to simulate and analyse the behavior of Real-Time Systems described by Timed Automata. WS-CDL is an XML-based language for describing Web Services, and Timed Automata are also represented in Uppaal by XML files, and thus, the translation has been developed by using XSLT, which is a language for transforming XML documents into other XML documents.

Keywords: RealTime Systems, Formal Methods, Verification, Web Services, Timed Automata, UPPAAL, model checking, translation, XSLT, WS-CDL, WST tool.

1 Introduction

Web Services are a key component of the emerging, loosely coupled, Web-based computing architecture. A Web Service is an autonomous, standards-based component whose public interfaces are defined and described using XML [1]. Other systems may interact with a Web Service in a manner prescribed by its definition, using XML [2] based messages conveyed by Internet protocols.

Web Services specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications. The future of E-Business applications requires the ability to perform long-lived, peer-to-peer collaborations between the participating services, within or across the trusted domains of an organization.

* This work has been supported by the CICYT project "Description and Evaluation of Distributed Systems and Application to Multimedia Systems",TIC2003-07848-C02-02 and the UCLM project "Aplicación de Métodos Formales al Desarrollo y Verificación de Web Services", with partial FEDER support.

Web Services cover a wide range of systems, which in many cases have strict time constraints, e.g., peer-to-peer collaborations may have time limits on completion. They are described in the top level layers in Web Services architectures [1] with elements such as *time-outs* and *alignments*. Time-outs allow each party to fix the time for an action to occur, while alignments are synchronizations between two peer-to-peer parties. Businesses rely on Web Services, it is therefore important that the Web Services framework ensure the correctness of systems with time constraints. Verification of real-time properties is especially important for Web sites that offer services with some kind of time restrictions, such as Web Services for control purposes, or even Web Services for business, we can think, for instance, in the maximum time to keep a reservation of a flight seat, or maximum delays to order fund transfers, etc. Since model checking of timed automata has proven to be very useful for similar problems in other areas, we investigate how it can be applied for Web Services.

This verification process starts from the top level layers of Web Services architectures [1] (Business Process Language Layer and Choreography layer). The particular Business Process Language layer that we use here is the Web Service Choreography Interface (WS-CI) [3], and the concrete Choreography Layer that we use is the Web Service Choreography Description Language (WS-CDL) [1]. Therefore, the starting point are specification documents written in WS-CDL and WS-CI. However, these description languages are not very useful for the verification process. Then, these descriptions are translated into timed automata, and the UPPAAL tool is used to simulate and verify the correctness of the system.

We illustrate the approach with a case study: a purchase by Internet, which contains some time constraints, and it is simple enough to illustrate the translation.

Related Work

There is a growing consensus that the use of formal methods, and the development of methodologies based on these formalisms, could have significant benefits in developing E-business systems, due to the enhanced rigor these methods bring.

Some of these approaches have used logics to formalize and reason about the contracts specified by the description languages [4], [5], [6] or [7]. They have the flexibility and expressive power of logics, but it is well known that in general such logics are not decidable, and thus verification requires expert knowledge. Closer to our approach are formalizations based on finite state machines [8,9] and Petri Nets [10], they allow automated verification using model checking tools; but they differ from our work in not treating timing constraints. Our approach is based directly on the theory of timed automata [11], and we use the UPPAAL tool [12,13,14], although other similar tools like KRONOS [15] would apply.

The paper is structured as follows. In Section 2 we describe the main features of the tool we use (UPPAAL), and then in Section 3 we show the WST tool makes the translation from WS-CDL into Timed Automata. In Section 4 we describe the main elements of the tool WST. The Case Study is presented in

Section 5, which illustrates the translation. Finally, conclusions and future work are presented in Section 6.

2 UPPAAL

UPPAAL [12] is a tool for modeling, validation and verification of real-time systems. The validation part is performed by graphical simulations and the verification part by Model-Checking means.

A system in UPPAAL consists of a network of concurrent processes, each of them modeled as a timed automaton. We assume the reader to be familiar with timed automata, so we omit a description of them in order not to enlarge the paper excessively. A detailed description of timed automata can be found in [16,17,12,18,19].

The simulation step consists of running the system to check that it works in a proper way in normal conditions. Since simulation does not guarantee completely the system correctness, we must use the verifier tool to check some properties of the system that are of interest, and that should be ensured in all conditions. For instance, we can check reachability properties, i.e. if a certain state is reachable or not. This is called Model-Checking and it is basically an exhaustive search that covers all possible dynamic behaviours of the system.

3 Translating WS-CDL documents to Timed Automata

Firstly, we are going to explain the steps that we have followed to get Timed Automata. As explained above, WS-CDL is an XML-based language for describing Web Services, and Timed Automata in UPPAAL are represented by another XML document, thus, the translation has been developed with XSLT [20], XML Stylesheets Language for Transformation, which is a language for transforming XML into other XML documents. Fig. 1 shows how the translation works. We have created XSL stylesheets, where we use XSLT instructions to extract the information from the WS-CDL document, and then to generate the elements which compose the UPPAAL document.

3.1 Correspondences between WS-CDL and Timed Automata

To obtain the translation we must see how the different elements of a WS-CDL description are translated into timed automata. The elements that we must consider are the following:

Role : They describe the behaviour of each class of party that we are using in the choreography. Thus, this definition matches with the definition of an automaton *template* in timed automata terminology.

Relation type : They define the communications between two roles, and the needed channels for these communications. In timed automata we assign a new channel for each one of these channels, which are the parameters of the templates that take part in the communication.

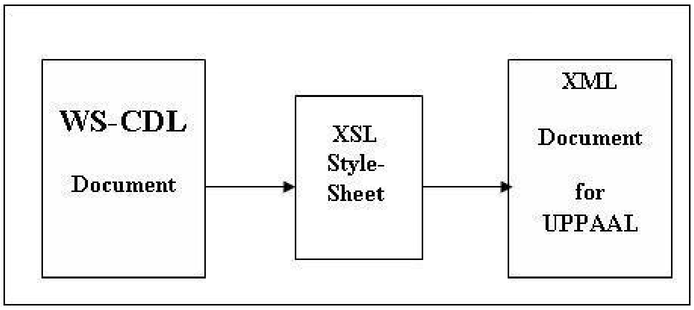

Fig. 1. Translation from WS-CDL documents to XML UPPAAL files.

Participant type : They define the different parties that participate in the choreography. In timed automata they are *processes* or automata participating in the system.

Channel types : A channel is a point of collaboration between parties, together with the specification of how the information is exchanged. As said before, channels of WS-CDL correspond with channels of timed automata.

Variables : They are easily translated, as timed automata in UPPAAL support (bounded integer) variables, which are used to represent information.

At this point it is necessary to define the behaviour of each template. This behaviour is defined by using the information provided by the flow of choreographies. Choreographies are sets of workunits or sets of activities. Thus, activities and workunits are the basic components of the choreographies, and they capture the behavior of each component. Activities can be obtained as result of a composition of other activities, by using sequential, parallel or choice compositions.

Finally, time restrictions are associated in WS-CDL with workunits and interaction activities. These time restrictions are introduced in timed automata by means of a local clock, guards and invariants. Therefore, when a workunit of an activity has a time restriction, we associate a guard to the edge that correspond to the initial point of this workunit in the corresponding timed automaton.

In Fig. 2 we can see the schematic presentation of the correspondence between WS-CDL and timed automata.

3.2 Generating the UPPAAL Document

The UPPAAL's XML file is structured in different sections, which are generated as follows:

- The header is the first section, which includes the version number, the type of document, and the URL of the DTD for UPPAAL systems.
- It follows the "Global declarations" section, where variables must be declared. The variables that we declare are the variables defined in the WS-CDL document, in the part called "VariableDefinitions". We declare channels, clocks, and integers.

Role	$\longrightarrow$ Template
Relation Type	$\longrightarrow$ Channel$^+$
Participant Type	$\longrightarrow$ Process$^+$
Channel Type	$\longrightarrow$ Channel
Variables	$\longrightarrow$ Variables
Choreography	$\longrightarrow$ Choreography$^+$ | Activity
Activity	$\longrightarrow$ Work Unit | Sequence | Parallelism | Choice
Sequence	$\longrightarrow$ Activity$^+$
Paralelism	$\longrightarrow$ Activity$^+$
Choice	$\longrightarrow$ Activity$^+$
Work Unit	$\longrightarrow$ State & Guard & Invariant
"`timeout`" of Interaction	$\longrightarrow$ Guard

Where the symbols +, | are BNF notation, and & is used to join information

Fig. 2. Schematic view of the translation

– Templates: As explained below, each distinct Role defined in the WS-CDL document is declared as a new template. Each template is defined with an automaton, which is generated with the information provided by the Choreographies.

  We have translated the different elements of the choreography into timed automata in the following way:

  1. Interaction Activities: For each interaction in the WS-CDL document, we have declared two nodes in the XML document. These nodes represent the source and the target element of the interaction. With the parameter "ChannelVariable" we have represented the synchronization between the collaborating parties. It is important to observe that interaction activities can define time constraints, depending on the optional parameter "timeout", in this case a "guard" must be declared.
  2. Sequence: A sequence is translated by concatenating the timed automata corresponding to the different activities that are included in the sequence structure in the WS-CDL document.
  3. Choice: This structure is translated by adding two nodes into the template. The first one is connected with each automaton performing an alternative activity included in the choice structure. The second node is included to represent the end of the choice structure.
  4. Workunit: It is translated by adding two nodes into the template. The first node represents the beginning of the workunit. This is connected with the initial node of the automaton that performs the activities inside the workunit. The second one is used to represent the end of the workunit structure. We have defined in the workunit time constraints with "guards" and "invariants". Fig. 3 shows the representation for the "choice" and "workunit" structures.

– The next section is the "Instantiation section", where processes can be declared as instances of a template.
– Finally, it comes the system definition. All the system Processes must be declared here.

Then, we have implemented this translation in the WST tool that we are developing (http://www.info-ab.uclm.es/fmc/tools/WS-CDLUppaal.rar), which is described in the next Section.

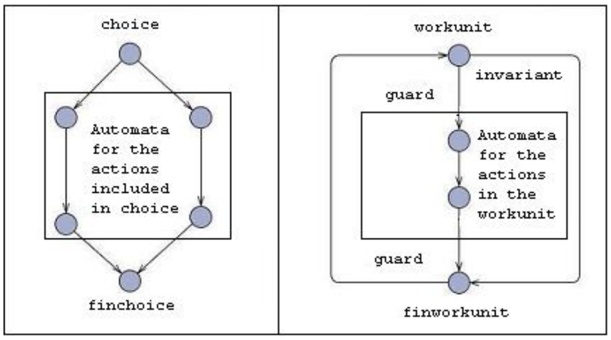

Fig. 3. Automata for the WS-CDL "Choice" and "Workunit" Activity

4 Web Services Transformation tool

WST tool, Web Services Transformation tool, which we have developed is an integrating environment for translating automatically WS-CDL specifications into Timed Automata. This tool applies XSL Stylesheets to an WS-CDL document, and the result is an XML document representing the system by Timed Automata.

In Fig. 4 a picture of the tool is shown, where in the left hand-side window you can see the WS-CDL document, and in the right hand-side the corresponding timed automaton (XML file).

The steps that we have to follow to transform a WS-CDL document into the corresponding XML UPPAAL documents are the following:

1. Firstly, opening the WS-CDL document. Once the file is opened, it can be visualized by the left hand-side window in the main interface of WST.
2. After that, clicking on the Transform button.
3. Once the translation is made, the result is shown in the right window of the WST tool.
4. Finally, the resulting XML UPPAAL document must be saved, in order to use it with the UPPAAL tool.

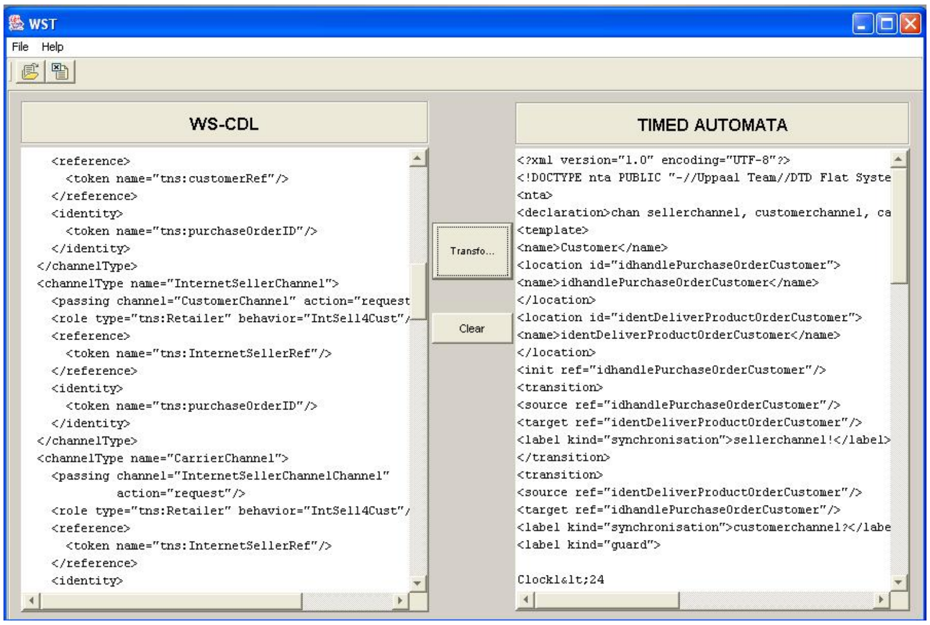

Fig. 4. A picture of the tool WST

As explained above, the obtained document as a result of the transformation can be used with the UPPAAL tool in order to validate the obtained system and verify the properties of interest.

5 Case Study

In this section, we are going to transform a WS-CDL document into Timed Automata using the Web Services Transformation Tool. The WS-CDL specification that we have translated corresponds to a system of purchases through Internet. The example that we use concerns the collaborative behaviour of the Customer, the Internet Seller and the Carrier.

When we open the WS-CDL document with the WTS tool, its contents are shown in the WS-CDL part, as we can see in Fig. 4. Then, we run the translation by clicking on the “Transform Button”, and the generated document with the transformation appears in the Timed Automata part of the WST, this system (represented by a XML document) can then be validated and verified with UPPAAL. Then, the obtained XML document can be opened with UPPAAL (Fig. 5 shows the corresponding UPPAAL window). As we can see, this system consists of three templates, which correspond to the distinct Roles defined in the WS-CDL document: Customer, InternetSeller and Carrier.

The automaton shown in Fig. 5 corresponds to the Customer process. This automaton has time constraints that are associated with the product delivery, which must be delivered in 24 hours (this constraint is represented by a guard).

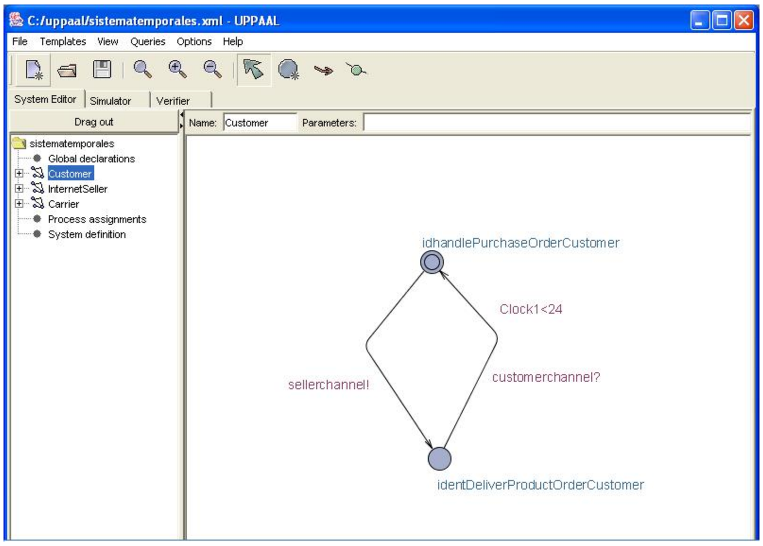

Fig. 5. Timed Automaton of the Customer (UPPAAL image)

Then, once we have a representation of our system in UPPAAL, we can use this tool to simulate and verify properties on it. For instance, we could check if the system is deadlock free, or if a certain state is reachable, or if a product is delivered in a given time, and so on.

6 Conclusions and Future Work

Web Services are widely used to implement a great variety of applications on Internet. Many of them have time restrictions, and then, it becomes important to verify that they will fulfill these restrictions. The translation of WS-CDL into Timed Automata allows us to verify with UPPAAL some properties of interest of the system, as well as to simulate it.

In this paper, we have seen that the specifications of Web Services applications can be translated into timed automata, and we have illustrated this translation by means of an example running in our tool. We have developed a tool, called WST, Web Services Transformation, in order to run this translation.

This is the first version of the WST tool, in following versions we are interested in implementing more functionalities, as the translation from WS-BPEL to Timed Automata. We are also working in the translation of RT-UML into Web Service Choreography Description Language, WS-CDL, in order to obtain WS-CDL documents from RT-UML diagrams corresponding to Web Services with strict real-time restrictions.

References

1. Nickolas Kavantzas et al. Web Service Choreography Description Language (WSCDL) 1.0. http://www.w3.org/TR/ws-cdl-10/.
2. Jean Paoli, Eve Maler, and et. al. Tim Bray. Extensible markup language (xml) 1.0 (third edition), 2004. http://www.w3.org/TR/2004/REC-xml-20040204.
3. Assaf Arkin et al. Web Service Choreography Interface (WSCI) 1.0. http:/www.w3.org/TR/wsci/.
4. H. Davulcu, M. Kifer, and I. V. Ramakrishnan. CTR-S: A Logic for Specifying Contracts in Semantic Web Services. pages 144–153, May 2004.
5. Aspassia Daskalopulu and T. S. E. Maibaum. Towards Electronic Contract Performance. In *DEXA Workshop*, pages 771–777, 2001.
6. A. Paschke, J. Dietrich, and K. Kuhla. A Logic Based SLA Management Framework. In 4th Semantic Web Conference (ISWC 2005).
7. Guido Governatori. Representing Business Contracts in *uleML*. *Int. J. Cooperative Inf. Syst.*, 14(2-3):181–216, 2005.
8. C. Molina-Jimenez, S. Shrivastava, E. Solaiman, and J. Warne. Contract Representation for Run-time Monitoring and Enforcement. Technical Report 810, University of Newcastle upon Tyne, School of Computing Science, Jul 2003.
9. H. Foster, S. Uchitel, J. Magee, and J. Kramer. Leveraging Eclipse for Integrated Model-Based Engineering of Web Service Compositions. In *ETX2005 Workshop at OOPSLA05*.
10. Aspassia Daskalopulu. Model Checking Contractual Protocols. *CoRR*, cs.SE/0106009, 2001.
11. Rajeev Alur and David L. Dill. Automata For Modeling Real-Time Systems. In *ICALP*, pages 322–335, 1990.
12. Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.
13. G. Diaz, F. Cuartero, V. Valero, and F. Pelayo. Automatic Verification of the TLS Handshake Protocol. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, 2004.
14. G. Diaz, K.G. Larsen, J. Pardo, F. Cuartero, and V. Valero. An approach to handle Real Time and Probabilistic behaviors in e-commerce: Validating the SET Protocol. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, 2005.
15. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In A. J. Hu and M. Y. Vardi, editors, *Proc. 10th International Conference on Computer Aided Verification, Vancouver, Canada*, volume 1427, pages 546–550. Springer-Verlag, 1998.
16. Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL: Status and Developments. Number 1254, pages 456–459. LNCS, Springer–Verlag, June 1997.
17. Fredrik Larsson, Kim G. Larsen, Paul Pettersson, and Wang Yi. Efficient Verification of Real-Time Systems: Compact Data Structures and State-Space Reduction. In *Proc. of the 18th IEEE Real-Time Systems Symposium*, pages 14–24. IEEE Computer Society Press, December 1997.
18. Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, Yi Wang, and Carsten Weise. New Generation of Uppaal. In *Int. Workshop on Software Tools for Technology Transfer*, June 1998.

19. Gerd Behrmann, Thomas Hune, and Frits Vaandrager. Distributed Timed Model Checking - How the Search Order Matters. In *Proc. of 12th International Conference on Computer Aided Verification*, Lecture Notes in Computer Science, Chicago, July 2000. Springer-Verlag.
20. James Clark. XSL Transformations (XSLT) Version 1.0. Technical Report REC-xml-19980210, W3C, 1998. `http://www.w3.org/TR/xslt`.