Modeling WS-BPEL with RT-UML Diagrams *

M.Emilia Cambronero J.José Pardo Gregorio Díaz Valentín Valero

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

Abstract. RT-UML is a UML Profile for modeling Real Time Systems, which can be used in particular to describe Web Services Orchestration with time constraints. For that purpose, we can use two classical UML diagrams, namely, the sequence and activity diagrams. Our goal in this paper is the design of Web Services with time restriction by using RT-UML. We introduce a translation of the main elements of RT-UML into Web Services Business Process Execution Language. In a previous works we presented a translation from WS-CDL and WS-BPEL into Timed Automata, which allows us to apply verification techniques and thus, to check some properties of interest.

1 Introduction

In the last few years some new techniques and languages for developing distributed applications have appeared, such as the Extensible Markup Language, XML [1], and some new Web Services frameworks [2,3,4] for describing interoperable data and platform neutral business interfaces, enabling more open business transactions to be developed.

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 [5,6]. Other systems may interact with a Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.

The context in which this work is situated is the development of a methodology for the generation and verification of "correct" Web Services. This metodology [7] establishes a top-down design for developing web services. The analysis phase is performed by a goal-model that specifies the properties that this kind of Real-Time systems must satisfy. Then, in the design phase we consider UML sequence and activity diagrams, in which we can capture the system time constraints. Between design and implementation we use a model-checking technique for discovering some errors and in this way, we improve the quality of design.

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

The source document for the implementation phase is a WS-BPEL document. Figure 1 shows this generation process. In this figure the RT-UML diagrams are translated into Web Services documents, which in their turn are also translated into Timed Automata in order to perform model checking on them.

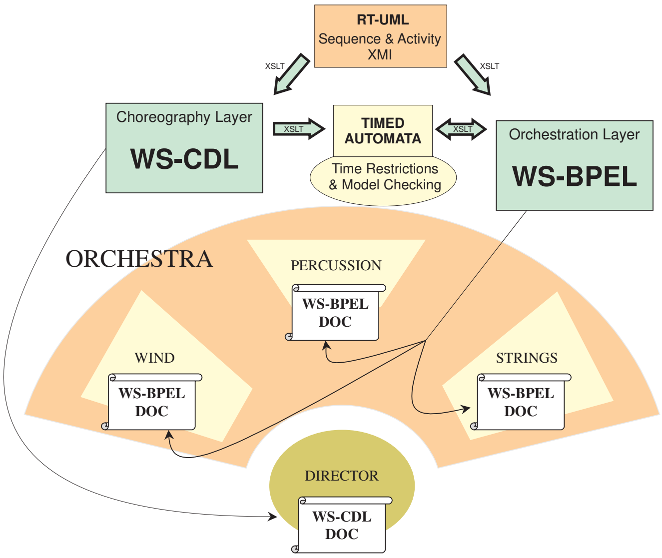

Fig. 1. The generation of correct Web Services

In [8,9,10] we have translated Web Services with time restrictions described by WS-CDL and WS-BPEL into timed automata, in order to simulate their behavior and verify some properties of interest. For that purpose, we used a well known tool that supports this formalism, UPPAAL [11] to simulate and analyze the system behavior.

We can find some related works in the literature: Mantell [12] proposes a methodology for translating UML activity diagrams into bpel4ws 1.0 Specification, and Skogan et al [13] propose another approach based on bpel4ws 1.1. In this paper we present a different vision, focusing our attention on the timed aspects of the systems. Hence, we are mainly concerned with Web Services with time constraints. It is important to highlight that we use both Activity diagrams and Sequence Diagrams, as well as the RT-UML Profile in order to model the time constraints of Real Time Web Services.

Most of the proposed composition languages for Web Services are based on XML [1], and although XML-based representations have their advantages as universal representations and exchange formats, they can be difficult to understand and to write for non-XML experts. Thus, the use of a graphical modeling language can be very useful to understand the behavior of Systems.

The motivation of this paper is the use of the Unified Modeling Language (UML) [14], and more specifically the RT-UML profile, as graphical modeling language for XML Real-Timed Web Services composition and the verification of these systems by using Model Checking techniques on Timed Automata. In other words, in this paper we focus our attention on the right part of Fig. 1, that is to say, the translation from RT-UML into WS-BPEL and from WS-BPEL into Timed Automata, this last part being shown in [8].

UML has been defined by the Object Management Group (OMG), the leading organization for object-oriented programming. RT-UML [15] allows us to see graphically the description of a Web Service business process, which helps us to understand the behavior of the participants in a business interaction; for that purpose the RT-UML diagrams are very useful, since they allow us to see this behavior in a clear way. More specifically, we use the sequence and activity diagrams for the description, since these diagrams can capture the main elements of a Web Services Business Process Execution Language and their time constraints. Afterwards, we then translate these diagrams into Web Services.

We have implemented transformation rules that can transform RT-UML models into WS-BPEL documents and these documents into Timed Automata, capturing, in every case, the time constraints. The method we propose uses RT-UML Sequence and Activity Diagrams to design Web Service compositions, and OMG's Model Driven Architecture (MDA [16]) to generate specifications in WS-BPEL and in Timed Automata.

As an illustration of this methodology, we use a particular case study, an Efficient Power Production Management System, whose description contains some time constraints.

The paper is structured as follows. In Section 2 we describe the main features of WS-BPEL. In Section 3 we present a brief description of the UML Profile for Real-time Systems, RT-UML, and we introduce the key elements of this profile, which is used in our translation to capture the System time constraints. The translation of RT-UML Sequence and Activity diagrams into WS-BPEL documents is presented in Section 4. In Section 5 we apply this methodology to the case study. Finally, the conclusions and future work are presented in Section 6.

2 WS-BPEL Description

The aim of the Web Services Business Execution Language, also known as Web Services Orchestration specification is to be able to describe precisely the behavior of any type of party and the collaboration with other parties, regardless of the supporting platform or programming model used by the implementation of the hosting environment. Using the Web Services Orchestration specification, a behavior is produced that contains a "specific" definition of the detailed ordering conditions and constraints under which behavior is performed, and which describes the specific internal behavior of the exchanged messages with all the parties involved.

In real-world scenarios, corporate entities are often unwilling to delegate control of their business processes to their integration partners. Orchestration offers a means by which the rules of participation within a collaboration can be clearly defined and agreed to, jointly. Each entity may then implement its processes as determined by the behavior described in the Orchestration document.

We will now introduce the reader to WS-BPEL. This is an interface description language that describes the observable behavior of a service by defining business processes consisting of stateful long-running interactions, in which each interaction has a beginning, a defined behavior and an end, all of this being modeled by a flow, which consists of a sequence of activities. The behavior context of each activity is defined by a scope, which provides fault handlers, event handlers, compensation handlers, a set of data variables and correlation sets.

Let us now see a brief description of these components:

- **Events**, which describe the flow execution in an event driven manner.
- **Variables**, which are defined by using WSDL schemes, for internal or external purposes, and are used in the message flow.
- **Correlations**, which identify processes interacting by means of messages.
- **Fault handling**, defining the behavior when an exception has been thrown.
- **Event handling**, defining the behavior when an event occurs.
- **Activities**, which represent the basic unit of behavior of a Web Service. In essence, WS-BPEL describes the behavior of a Web Service in terms of choreographed activities.

3 UML Profile for Schedulability, Performance, and Time

In this section, we introduce the key elements of the profile that we have used in our translation, but first it is important to use this profile to model Real-time Systems.

The use of this profile [15] is justified because UML is lacking in some key areas that are of particular concern to real-time system designers and developers. In particular, the lack of a quantifiable notion of time and resources was an impediment to its broader use in the real-time and embedded domain. It was discovered that UML had all the requisite mechanisms for addressing these issues, in particular through its extensibility faculties. The UML Profile for Schedulability, Performance, and Time is a standard way of using these capabilities to represent concepts and practices from the real-time domain.

One of the main guiding principles is that modelers, as far as possible, should not be hindered in the way they use UML to represent their systems in order to be able to do model analysis. That is, rather than enforcing a specific approach or modeling style for real-time systems, the profile should allow modelers to choose the style and modeling constructs that they feel are the best fit for their needs of the moment.

In this paper we have chosen the "General Time Modeling" viewpoint to model Real-time Systems, because this model describes a general framework for representing time and time-related mechanisms that are appropriate for modeling real-time software systems. It is quite general, but a given application only needs to use a subset of the concepts and semantics required. We have then chosen a set of the concepts of this viewpoint to model our real-time Web Services.

The concepts of RT-UML that we have used in this paper to capture the time constraints of Web Services are the following:

- TimedAction: This very general and very useful concept is modeled by applying the <<RTaction>> stereotype to any model element that specifies an action execution or its specification (which is a way of defining defaults for instances of those specifications). This includes action executions, methods, actions (including entry and exit actions of state machines), action states, subactivity states, states, and transitions. It can also be applied to stimuli that take time to arrive at their destination.

  Timed Actions have associated the following tagged values:

  duration: The time interval in which the action occurs.
  start: The time of the event occurrence when the action started.
  end: The time of the event occurrence when the action was completed.

  The start and end times of the action are specified by appropriate tagged values (RTstart and RTend respectively). Alternatively, they may be tagged with the RTduration tag. The two forms are mutually exclusive.
- Delay: A kind of timed action execution that represents a null operation for a pre-specified time interval. This action has no side-effects except to delay the action execution that follows it. This is modeled by a model element that is stereotyped as <<RTdelay>>. It can only have an RTduration tag associated with it. Delays can be placed on the same model elements as timed actions.
- TimedEvent: This very general and very useful concept is modeled by applying the <<RTevent>> stereotype to any model element that implies an event occurrence, and it models any event that occurs at a known time instant.

  The TimedEvent only uses the tag value "RTat", indicating the instant at which the event occurs.
- Clock: A Clock is a kind of timing mechanism that generates a clock interrupt periodically, where a TimingMechanism is an abstract concept that captures the common features of resources that specialize in performing time measurement and timing-related functions.

  Clocks are modeled by model elements that are stereotyped as <<RTclock>>. An instance of a clock can be identified using the "RTclockId" tag.
- reset(): An operation that stops the timing mechanism and sets it back into its initial state. The stereotype <<RTreset>> models the reset() operation on a timing mechanism.

Object	⟶Process
Messages	⟶Partnerlinktypes + portTypes + message + singleMessage
singleMessage	⟶receive | request| reply
Variables	⟶Variables
Expressions	⟶Xpath expresions | Assign activity
Activities	⟶Ordering structure Sequence$^+$ | Activity
Decision Points and Merge Construct	⟶Structured activity if
Concurrent Activities	⟶wait activity + Structured activity Flow + wait activity
Guards	⟶Condition of Activity if
Swimlane	⟶Ordering structure Sequence$^+$
RTaction	⟶Activity with scope definition
RTevent	⟶wait activity + activity
RTdelay	⟶wait activity
RTclock	⟶Variable (clock)
RTreset	⟶assign activity

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

Fig. 2. Schematic view of translation

4 Translation from RT-UML Diagrams into WS-BPEL

In this section we show the translation of the RT-UML sequence and activity diagrams into WS-BPEL XML format. The first subsection explains the translation for Sequence Diagrams, the second one shows the translation of activity diagram elements and the third one explains the tranlation of real-time elements. These translations have been developed with XSLT [17], XML Stylesheets Language for Transformation, which is a language for transforming XML into other XML documents. In the market we can find some tools that allow us to model the RT-UML diagrams and to obtain the corresponding XMI document, as Artisan 6.0 tool [18], this document can be easily translated to the WS-BPEL XML documents by using XSLT. In Table 2 we can see a schematic view of the translation.

4.1 Translation from RT-UML Sequence Diagrams into WS-BPEL

The system is designed by using RT-UML activity and sequence diagrams. But notice that we just need to use a single sequence diagrams, representing the whole system.

The translation of the main elements of RT-UML sequence diagrams into WS-BPEL is as follows:

- **Messages** : The messages in RT-UML Sequence diagrams correspond to the following entities of WS-BPEL:

- PartnerLinkTypes: The partnerLinkTypes represent the interaction between a service and each of the parties with which it interacts. The main utility of PartnerLinkType is to characterize the relationship between two services, this is possible by defining the **roles** played by each of the services that participate in the conversation and specifying the **portType** provided by each service to receive messages.

  Each role specifies exactly one WSDL portType.

– **Objects** : The objects in UML Sequence diagrams can be represented by the following entities in WS-BPEL:

- Processes: In WS-BPEL we can find two types of processes, the execution process, and the abstract processes. Execution processes model the behavior of each particular participant in the interaction, while abstract processes are meant to couple Web Service interface definitions with behavioral specifications that can be used both to constrain the implementation of business roles and to define in precise terms the behavior that each party in a business protocol can expect from the others. Then, the translation of the RT-UML objects is made only by using the execution processes, since they allow us to model the behavior of each particular participant in an interaction.

  From the RT-UML description we obtain the needed elements to complete the WS-BPEL process definition; more precisely:

  1. Variables section: here the variables identified in the RT-UML document must be declared.
  2. PartnerLinks section: in this section the different parties that interact with this particular business process are declared. This information can be easily extracted from the RT-UML document. Notice that each partnerlink must have a partnerlinktype (that models the interaction between the parties) and a role name, which is the other participant in the interaction.
  3. FaultHandlers section: where the activities that must be performed in response to faults resulting from the invocation of the assessment and aproval services. The information contained in this section is extracted from RT-UML frames, and more specifically, with the "opt" and "alt" labels.
  4. EventHandlers section: here the activities that must be performed in response to events or alarms are described, and this information is taken from the RTevent of RT-UML.

– **Expressions** :

RT-UML expressions are immediately translated into WS-BPEL expressions, but the needed variables must be first declared in WS-BPEL. Notice that we should use XPath 1.0 to introduce the expressions in WS-BPEL.

Concening with assignments, we can use the assign activities of WS-BPEL to copy data from one variable to another, to construct and insert new data using expressions.

4.2 Translation from UML Activity Diagrams into WS-BPEL

Let us now see the translation of the main elements of RT-UML activity diagrams into WS-BPEL.

- **Activities** : An activity on a UML activity diagram typically represents the invocation of an operation, a step in a business process, or an entire business process.
  Single activities are immediately translated into basic activities of WS-BPEL. More complex activities of RT-UML will be sequential (in fact, there will be an initial and a final activity), and they will be translated by using the *sequence* structured activity of WS-BPEL.
- **Decision points and Merge construct** : in RT-UML they are modeled by using diamonds. They are translated into WS-BPEL *if* structured activities. In fact, we can have an "elseif" element, and an "else" element.
- **Guards** : The guards of RT-UML are translated by using the *condition* part of the "if" activities in WS-BPEL.
- **Concurrent activities and synchronization**:
  In RT-UML Activity Diagrams the starting point of these parallel activities is represented by a horizontal split, which is called a fork, and it has one transition entering it and two or more transitions leaving it. There is also another horizontal line that represents the end of the parallel activities, with two or more transitions entering it and only one leaving it. In WS-BPEL the concurrency and synchronization between activities is provided by the *flow* Structured Activity. Within the flow structured activity we must indicate the activities that are made in parallel.
  To capture the initial synchronization among the involved activities we use a *wait* basic activity, as well as to synchronize the end of all these activities.
- **Swimlane guidelines** : This is a way to group activities performed by the same actor on an activity diagram.
  In WS-BPEL we can use the Ordering structure *Sequence to represent the Swimlane guidelines of RT-UML Activity Diagrams.*

4.3 Translation of Real-Time elements of RT-UML diagrams

RT-UML introduces three new elements in the Sequence diagrams to capture some important elements of real-time systems such as delays, events and actions.

The translation of these elements works as follows:

- **RTDelay** represents a delay for some units of time. In WS-BPEL we can use the *wait* activity to indicate this delay.
- **RTEvent** is used to indicate an event which occurs at a known time instant. To translate this element to WS-BPEL we use again a *wait* activity followed with the activity associated with RTEvent.
- **RTAction** is a timed action having a duration (alternatively you can indicate a starting and finishing time).

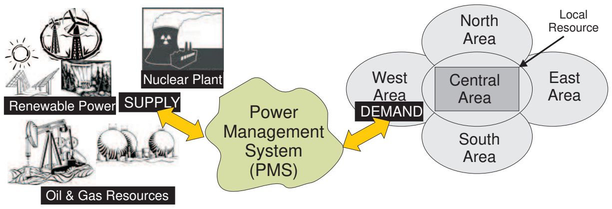

Fig. 3. The power supply versus the customer demand.

We use a *scope* element to translate this kind of action. In this scope we define an *onalarm* event, we translate the duration of the activity by using the *for* part within *onalarm*, which allows us to establish this duration.

- **RTreset** changes a clock value to zero, then it is immediately translated in WS-BPEL by using an *assing* activity.
- **RTclock** is a clock in RT-UML, and it is translated by using XPath variables.

5 Case Study: Efficient Power Production Management System

This example defines the general characteristics for an efficient power production management system with the basic restriction that the electric power is *non-storable*. This restriction implies that the power supply must be adapted to the demand. This example is based on the management system of the Spanish power lines.

In this system, we have considered three different types of electric power resources: Renewable power, pollution power and high-risk power. The renewable power plants are wind turbines, solar plants and hydroelectric plants. The pollution power plants are oil and natural gas plants. And the high-risk resources are nuclear plants.

Furthermore, the power demand is divided into areas: Central, north, south, east and west. The central area is a location where a set of resources is settled, whereas, the neighboring areas are not directly connected with electric power resources.

This system is depicted in Figures 3 and 4. The power management system described in Fig. 4 shows a pyramidal system for attending and managing the demand versus the supply. The restrictions of this system consist of two types of general restrictions and time restrictions, and are defined as follows:

- **General Restrictions**
  1. *Hierarchical demand* . Two levels for managing the demands, firstly local and secondly central.
  2. *Environmental restrictions.* If pollution environmental sensors are activated, then only renewal and high-risk power resources are available.

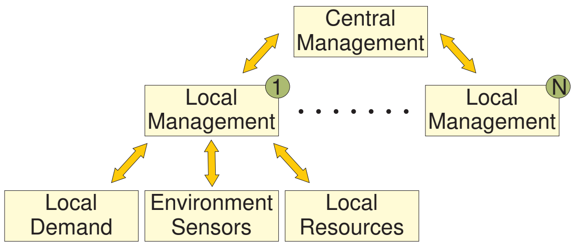

Fig. 4. The power management system (PMS).

Furthermore, these sensors inform of the amount of power that the renewal resources could supply.

3. *Hierarchical supply.* The priority for selecting the resources is: firstly renewable resources, secondly pollution resources and thirdly high-risk resources.

– **Time Restrictions**
  1. The demand is calculated 12 hours before.
  2. The local management takes an average of one hour to process a demand.
  3. Renewable resources take two hours in order to supply the power, pollution resources take one hour and high-risk resource take five hours.
  4. The supply for adjacent areas takes one hour over the above times.
  5. The central management takes one hour to process a demand, as the local management.

Figure 5 shows a particular RT-UML sequence scene of this system. This scene captures the situation for attending the demand of a specific area. This request is performed by a "demand" message, which indicates the area in which the source of the demand has been generated and the total power amount that will be necessary in 12 hours. The destination of this request is the local management, which has one hour to decide if the "local resources" have enough power to attend the demand (depicted in Fig. 5 with the labeled frame "choice1") or to send the demand to the "central management" (depicted in Fig. 5 with the labeled frame "choice2"). This decision is performed by dealing with two factors: environmental factors and resources availability, as mentioned previously in the restrictions. If the decision is to perform "choice1", then the "local management" sends "supply" messages to the corresponding local resources, which indicate the type and amount of power that must be generated in order to attend this demand. However, if the decision is "choice2", then the "local management" informs the "central management" by sending a "demand" message with the requested amount. The "central management" has one hour to deliver this request to another "local management", in which there are available resources to attend the demand or part of this demand. Note that this information is available due to the "local management" informing the "central management" with "supply" messages, which contain information about the availability of resources and

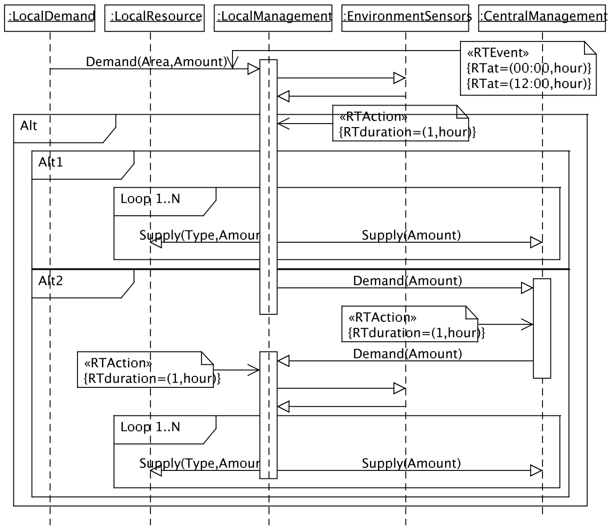

Fig. 5. Sequence diagram for the PMS.

the amount of power that is going to be generated. Figure 6 shows an activity diagram for the "local management" described in this sequence scene.

Figure 7 shows a part of the WS-BPEL specification for a "local management" process, which has been obtained by applying the transformation rules from the previous section. This piece of the specification corresponds to the situation in which the "local management" receives a "demand" request from the "local demand". For instance, in this specification, we can see how the RT-Event "demand" is transformed into a wait-until structure in which a receive message is specified and the decision performed between "choice1" and "choice2" is transformed into an **if** control structure. For a better readability, we have replaced some Xpath expressions, which are too complex and large, with natural language sentences.

6 Conclusions and Future Work

In this paper we have presented a translation of RT-UML Sequence and Activity diagrams into WS-BPEL documents. Sequence diagrams precisely describe the collaboration and message flow between the parties involved in a choreography, and also capture the time windows in which the conversations can take place. On the other hand, activity diagrams describe in detail the way in which the different actions are done, by means of sequence, choice, parallelism and time outs.

The descriptions of Web Services by WS-BPEL documents, written in XML format, can be very difficult to understand for non-experts, so we use RT-UML in order to design these Real Time Web Services.

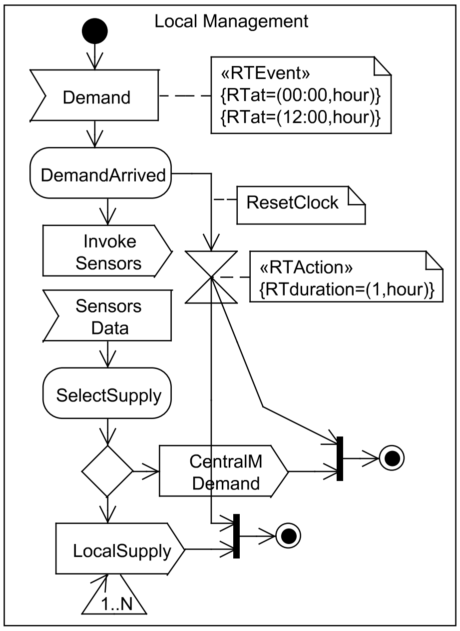

Fig. 6. Activity diagram for a "Local Management".

Our methodology can exploit some previous works, in which both WS-CDL and WS-BPEL descriptions are translated into timed automata, thus allowing us to simulate and analyze the system behavior, for instance by using the UPPAAL tool.

Observe that the automatic generation of WS-BPEL documents, which are very close to an implementation, provides us with a correct design, at least with respect to the RT-UML Diagrams of reference.

The translation presented is therefore a powerful tool for obtaining correct Web Services. We have established the equivalence between the different elements of the RT-UML sequence and activity diagrams with the elements of the WS-BPEL specifications, as a design method, and the translation of these WS-BPEL documents to Timed Automata, in order to perform Model Checking on them. Furthermore, these translations have been illustrated with an example.

Our future work addresses the issue of developing a tool capturing all these capabilities: RT-UML modeling, translation to WS-BPEL and WS-CDL, as well as the generation of timed automata XML files. We have a preliminary version of the tool that can be found in the URL http://www.info-ab.uclm.es/fmc/tools/WS-CDLUppaal.rar.

Fig. 7. WS-BPEL process specification of a Local Management party.

References

1. 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.
2. Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers. Uddi version 3.0.2, 2004. http://uddi.org/pubs/uddi_v3.htm.
3. Marc Hadley, Noah Mendelsohn, and et. al Jean-Jacques Moreau. Soap version 1.2 part 1: Messaging framework, 2003. http://www.w3.org/TR/soap12-part1.
4. Sanjiva Weerawarana, Roberto Chinnici, and et. al. Martin Gudgin. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, 2004. http://www.w3.org/TR/2004/WD-wsdl20.
5. Assaf Arkin, Sid Askary, and et. al. Ben Bloch. Web Services Business Process Execution Language Version 2.0, December 2004. http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm.
6. Nickolas Kavantzas et al. Web Service Choreography Description Language (WSCDL) 1.0. http://www.w3.org/TR/ws-cdl-10/.
7. G. Diaz, M. E. Cambronero, M.LL. Tobrarra, V. Valero, and F. Cuartero. Analysis and Verification of Time Requirements Applied to the Web Services Composition. In *Proceedings of WS-FM, Viena*, Lecture Notes in Computer Science, pages 178–192. Springer, September 2006.

8. G. Diaz, M. E. Cambronero, J. J. Pardo, V. Valero, and F. Cuartero. Automatic Generation of Correct Web Services Choreographies and Orchestrations with Model Checking Techniques. In *Proceedings of International Conference on Internet and Web Applications and Services ICIW'06*. IEEE Press.
9. G. Diaz, J. J. Pardo, M. E. Cambronero, V. Valero, and F. Cuartero. Verification of Web Services with Timed Automata. In *Proceedings of First International Workshop on Automated Specification and Verification of Web Sites*, Electronic Notes in Theoretical Computer Science.
10. G. Diaz, J. J. Pardo, M. E. Cambronero, V. Valero, and F. Cuartero. Automatic Translation of WS-CDL Choreographies to Timed Automata. In *Proceedings of WS-FM, Versalles*, Lecture Notes in Computer Science, pages 230–242. Springer, September 2005.
11. 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.
12. Keith Mantell. From UML to BPEL, September 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel.
13. David Skogan, Roy Grønmo, and Ida Solheim. Web service composition in uml. In *EDOC*, pages 47–57, 2004.
14. OMG. *UML 2.0 Superstructure proposal v.2.0.*, January 2003.
15. UML Profile for Schedulability, Performance, and Time Specification, Version 1.1. In http://www.omg.org/docs/smsc/04-12-05.pdf.
16. David Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, Inc., New York, NY, USA, 2002.
17. James Clark. XSL Transformations (XSLT) Version 1.0. Technical Report REC-xml-19980210, W3C, 1998. `http://www.w3.org/TR/xslt`.
18. Artisan tool 6.0. *Artisan Real Time Studio*, 2001.