Real-time Web Services Orchestration and Choreography

Kawtar Benghazi, Manuel Noguera, Carlos Rodríguez-Domínguez, Ana Belén Pelegrina, and José Luis Garrido

University of Granada
Department of Languages and Computer Systems
GEDES Research Group
C/ Periodista Daniel Saucedo Aranda S/N. 18071 Granada, Spain
{benghazi,mnoguera,carlosrodriguez,abpelegrina,jgarrido}@ugr.es

Abstract. Real-Time issues are not usually considered when describing and composing web services. However, modern web services are usually involved in the software implementation of time-constrained business processes [1]. The satisfaction of time constraints is crucial in order to ensure the validity of systems where the response to a certain stimulation has to take place in a shortened period of time. Thus, the service composition problem becomes more complex, since time restrictions should be taken into account both in the choreography and orchestration processes in order to establish the temporal consistency of the web services. In this paper, we present a formal approach for real-time service orchestration and choreography. In this regard, we use UML-RT as a visual and user-friendly notation in order to model services and their interactions, Timed CSP as an underlying formal grounding to enable services verification and WS-BPEL as an execution language.

Key words: Timed-web services, web-services composition

1 Introduction

Web services are loosely-coupled modular software applications that interact with each other through web technologies. In order to enable inter and intra organizational integration, these web services should be composed to work together to carry out the integrated business process goals.

The choreography and orchestration are important emerging mechanism that deal with the problem of web services composition. Choreography is concerned with the web services interaction coordination, and orchestration is concerned with the creation of high level web services (called orchestrators) from existing ones.

Whenever web services are involved in the implementation of time-constrained business processes [1], their composition becomes very complex, since time restrictions should be taken into account both in the choreography and orchestration processes in order to establish the temporal consistency of the web services.

J. Barjis, M.M. Narasipuram, G. Rabadi, J. Ralyté, and P. Plebani (Eds.):
CAiSE 2010 Workshop EOMAS'10, Hammamet, Tunisia, pp. 142-153, 2010.

To this matter, it is important to introduce the notion of *timed web services*, which are services whose behavior and interactions with other services have to accomplish pre-established time restrictions.

Capturing time restrictions in orchestration and choreography requires the usage of languages with enough expressive power to represent complex relations about timely interaction between services. Moreover, it is very important to use formal methods with a well defined syntax and semantics in order to ensure the *compositionally* of timed web services and in order to verify the temporal properties that they should fulfill. Several authors (e.g., [2]) have advocated for the usage of process algebra for describing services and for reasoning on their properties. In this paper, we use Timed CSP [3] for the orchestration and choreography of timed web services. Timed CSP has several characteristics that make it suitable in order to describe both the behavior and the interactions of timed web services:

- It allows the description of temporal constraints.
- It has a well defined denotational and operational semantics [4]. On that basis, it can be checked whether processes satisfy properties of models by means of model checker tools (e.g., HORAE [5], FDR [6]) or *bisimulation*
- Compositionally: the denotation of a process is constructed from denotations of its parts.

Timed CSP and formal methods have the disadvantage of being tedious to use. In order to overcome this problem we combine this language with UML notations, specifically with timed sequence diagrams [7] and timed state diagrams [7]. Thus, both the choreography and the orchestration of timed web services are carried out in a visual manner and, then, are mapped to Timed CSP processes by applying a set of transformation rules.

In the field of web services, the Web Service-Business Process Execution Language (WS-BPEL) is a commonly adopted standard with a rich set of constructs to compose services. In this paper, we establish a set of mapping rules to systematically derive code from Timed CSP processes to WS-BPEL.

Several works in the literature have addressed the composition of web services [8]. In [9], sequence diagrams are used for the choreography of web services. Also, the transformation from sequence diagrams to WS-BPEL is given. In [10], UML-RT is used in order to describe timed web services. Other Works use formal methods for describing and reasoning on web services. For example, [11] [12] uses petri nets, [2] and [13] use CCS and [14] uses π-calculus. However, the time constraints and the temporal consistency between services was not addressed in any of these approaches.

This paper is organized as follows: Section 2 gives an overview of Timed CSP language. In Section 3, the timed web services and its constituents are defined. Both Section 4 and Section 5 describe proposed web services composition techniques. In Section 6, a mapping from Timed CSP to WS-BPEL is defined. Finally, brief conclusions drawn from this paper and some ongoing work is described in Section 7.

2 An Overview of Timed CSP

Timed CSP has been proposed as a specification language in order to model and to reason on concurrent, parallel systems that must fulfill explicit time constraints. Timed CSP in an extension over CSP [15] that introduces the capability of quantifying temporal aspects of sequencing and synchronization [16]. In this paper, Timed CSP terms are constructed according to the following grammar:

$$P ::= Stop \mid Skip \mid Wait\ t \mid a \rightarrow P \mid P; Q \mid$$
$$P \Box Q \mid \ P \overset{t}{\triangleright} Q \ \mid P \parallel Q$$

These terms has the following intuitive interpretations:

- P is a process.
- $Stop$ is a deadlocked, stable process which will never engage in any external communication. It is the "broken program".
- $Skip$ does nothing except to end successfully $\surd$. This process is equivalent to $\surd \rightarrow STOP$.
- $Wait\ t$ does nothing, but it is ready to end after a delay t.
- $a \rightarrow P$ is a process that is initially prepared to engage in an event a, and then it behaves as P.
- $P; Q$ corresponds to the sequential composition of P and Q. It represents a process that behaves like P until P chooses to terminates and becomes to behave like Q. The process Q is turned on at the exact moment that P terminates.
- $P \Box Q$ corresponds to a process that is willing to behave like either P or Q. The choice is made by the environment. The decision is taken on the first visible event (The process is non deterministic only if the first visible event of P and Q are equal).
- $P \overset{t}{\triangleright} Q$, is the process that initially gives P the priority of turning on. If no visible event from P has occurred in t units of time, the process behaves as Q, and P never turns on.
- $P \parallel Q$ represents a parallel composition of processes P and Q.

A variety of semantic models were defined for the Timed CSP language.

The Timed CSP *operational semantic model* is given in terms of two relations: an evolution relation, which describes the situation in which a process becomes another one by simply allowing time to pass, and a timed transition relation, which describes when a process becomes another one by performing an action at a particular time.

The *denotational model* is used in order to provide formal semantics to Timed CSP terms. It also allows to specify the required behavior of a process, that is, its desired properties.

The properties to be satisfied by a system or a process are defined in terms of timed failures (or timed traces). This definition characterizes some timed failures (timed traces) as acceptable and some others as non-acceptable. A process

complies with its specification only if all its executions are acceptable, that is, none of its executions violates its specification.

Likewise, if we denote $(tr, \aleph)$ as a timed failure and $S(tr, \aleph)$ as a predicate in the timed failure, then it is said that P $satisfies$ or $complies$ with $S(tr, \aleph)$ if $S(tr, \aleph)$ holds for any timed failure (trace) of P.

$$P \textbf{ sat } S(tr, \aleph) \Leftrightarrow \forall (s, \aleph) \in tfailure(P) : S(tr, \aleph)$$

The specification of a process in terms of timed traces allows the description of both *safety* and *liveness* properties [17].

3 Real-Time Service Elements

Services are self-contained active entities. Real-time services are services that must behave under certain time restrictions. Thus, the operation of a real-time service can only be considered correct if it delivers a valid result in a predefined time range.

A real-time service is conceptually equal to a *capsule* in UML-RT. Thus, in this paper it is used the same visual representation of an UML-RT capsule to represent real-time services. Figure 1 shows the elements involved in a communication with services.

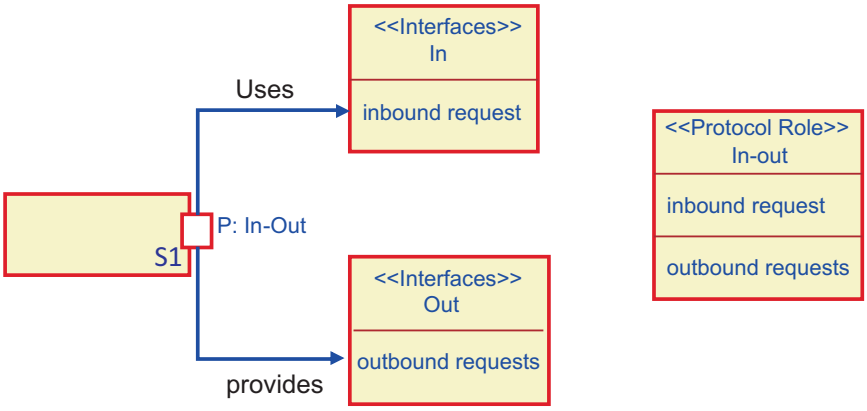

Fig. 1. Service, port, interfaces and protocol role.

A real-time service S_i communicates with other services through a set of interaction ports P_k and according to a pre-established communication protocol. It is important to mention that ports in our approach are active and therefore they have an associated behavior.

Each port can be associated to one or two interfaces. This interfaces represents access points that control the behavior of the whole service. We can distinguish between two types of interfaces:

- *Provided interfaces* that describe the set of input requests and represent the set of public operations provided by the service.
- *Required interfaces* that describe the set of output requests and represent the set of public operations required by the service.

The set of input and output requests are encapsulated in protocols and protocol roles.

Definition 1: Real-time services Real-time services are modular software applications that interact with each other through interfaces. This definition coincide of the definition of a process in Timed CSP conceptual framework. Thus, a real-time service can be represented by a process P in Timed CSP.

A real-time service has an internal and an external timed behavior:

- *Internal timed behavior*. It describes the behavior of the service that is associated with the accomplishment of its internal tasks. This behavior can be represented as a set of timed events that may be observed during the service execution. These are the timed traces $\sigma = \langle(\sigma_1, t_1), (\sigma_2, t_2) \dots (\sigma_n, t_n)\rangle$ with $\forall i \in 1..n$, $\sigma_i \in \alpha(P)$, $\alpha(P)$ the set of all action of P and t_i the time occurrence of the event σ_i.
- *External timed behavior*. It describes the set of interactions between a service and its environment (i.e., a system composed by other services). Moreover, it provides a *black box* that only shows an abstraction over the functionality that is provided by a service, which will also hide its internal actions. Formally, the internal behavior σ_{int} of a timed-service can be deduced from its internal one by hiding all its internal actions: $\sigma_{int} = \sigma \setminus INT$, with INT a set of internal action of a service.

It is possible to distinguish two types of real-time services:

- *Primitive services* are services that are not able to be divided.
- *Composite services or orchestrators* are a set of services that are composed of existing services.

4 Real-time Service Choreography

The choreography is concerned with the interaction between services. This interaction is carried out through a set of messages exchanged between the interfaces or the ports of the services. Due to this message passing nature, undesirable situations such as deadlocking can be reached if the behavior of the services is inconsistent. In order to ensure behavior consistency between communicating services, the communications between their interfaces or ports must be coordinated.

Definition 2: Business Protocol The business protocol represents a valid communication sequence (conversation) and can be specified by a timed sequence diagram (as it is shown in figure 2) or its corresponding timed traces (by applying the rules described in [18]) and a set of temporal restrictions. Each trace has the form $\langle (e_1, t_1), (e_2, t_2) \ldots (e_n, t_n) \rangle$, where each t_i represents the instant of time when the corresponding event e_i occurs and, therefore, it establishes a partial order in the occurrence of events. The set of the temporal restrictions has the form $\{t_i \; Op \; t_j[+d]\}$, being Op a relational operator and d an optional amount of time.

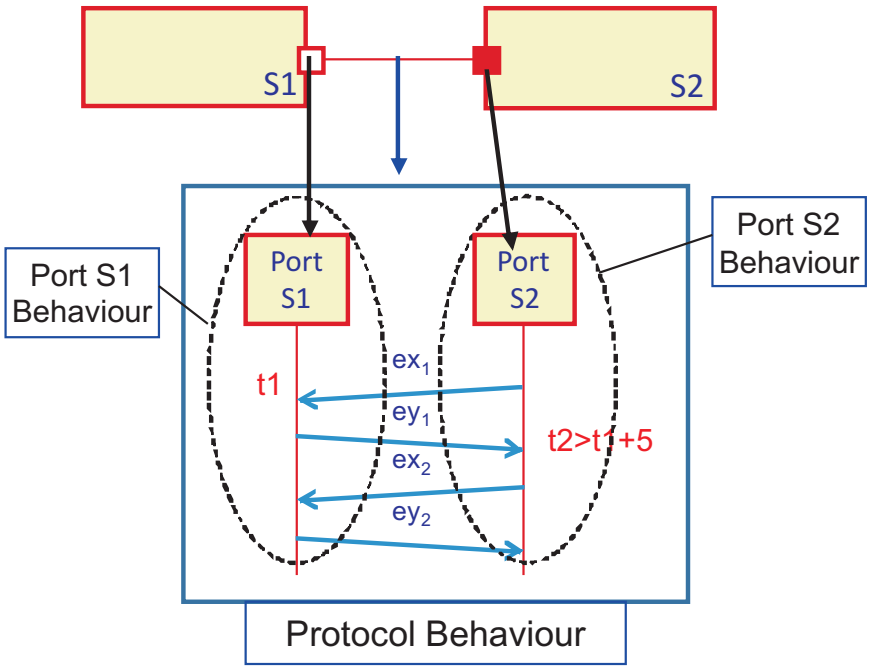

Fig. 2. Services choreography using Timed sequence diagram.

Definition 3: Deadlock free A communication between two services σ_1 and σ_2 is deadlock free if the parallel composition of the two services preserves the execution order of the messages and the temporal constraints specified in a conversation SD. Formally,

$$\sigma_1 || \sigma_2 \models_T SD$$

As an application example, let us consider a wire transfer order in an on-line banking system. Once the client has confirmed all data and has ordered to proceed through his/her web client, the transfer validation web service at the

bank server generates a random key and sends it out to the client's mobile phone by invoking another web service of the client's mobile network operator. It also randomly selects one position in the matrix of secret codes that every client has and that each branch office provides to its clients. We call that value the code card matrix position (CCMP). Then, the transfer validation service requests the web client for the two codes: the one randomly generated and sent to the client's cell phone and the one from the client's code card. This process has to be completed in 15 seconds, from the time the proceed order is received until the time an SMS is delivered to the client's cell phone. Afterwards, the client has 150 seconds to provide the requested codes. Otherwise the transfer would be canceled. The interaction between the transfer validation and the web client services is described in figure 3.

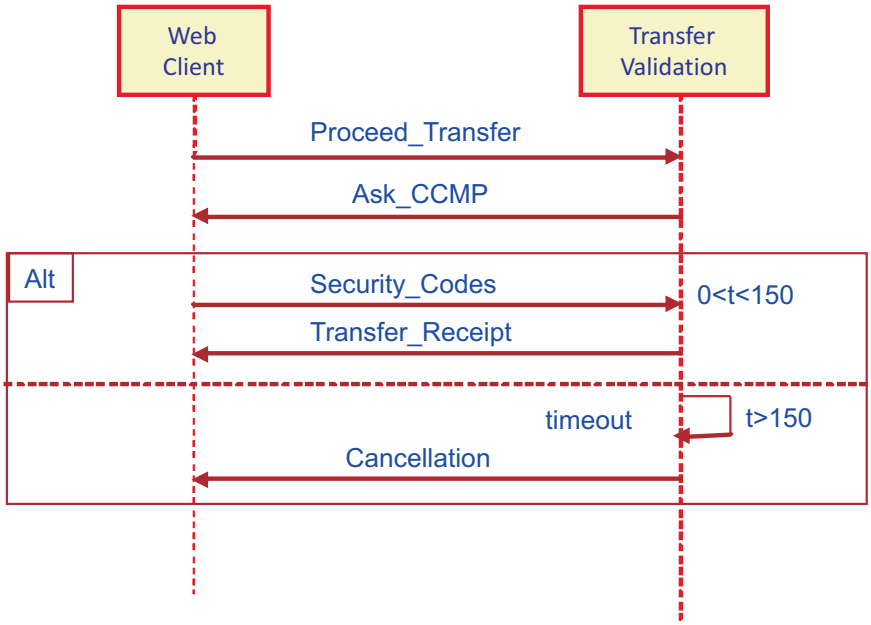

Fig. 3. Timed sequence diagram

5 Bottom-Up Service Orchestration

In our approach, the internal behavior of primitive services is modeled by transforming its timed sequence diagram into a corresponding timed state diagram, following a set of rules established in [18]. A specification of this behavior in Timed CSP syntaxis can be derived from a timed state diagram. Figure 4 shows

the internal behavior of the web client and the transfer validation services that were introduced in previous section.

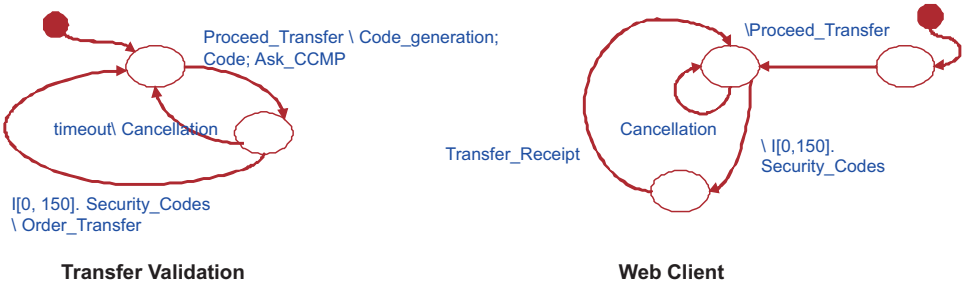

Fig. 4. Internal behavior of web client and transfer validation services

By deriving timed state diagrams in figure 4 to Timed CSP, it is obtained the formal specification of the transfer validation service (annotated as TV) and the formal specification of the client (annotated as WC) [1]:

$$TV = C1?Proceed\_Transfer \rightarrow Code\_Generation \rightarrow C2!Code \rightarrow C1!Ask\_CCMP)$$
$$(C1?Security\_Codes \rightarrow (C1!Transfer\_Receipt \rightarrow SKIP))$$
$$\overset{150}{\triangleright} (C1!Cancellation \rightarrow SKIP)$$
$$WC = C1!Proceed\_Transfer \rightarrow (C1!Security\_Code \rightarrow C1?Transfer\_Receipt \rightarrow$$
$$SKIP) \overset{150}{\triangleright} (C1?Cancellation \rightarrow SKIP)$$

The specification of composite services or orchestrators is deduced from the formal specification of its components (i.e, existing services) following a bottom-up process, as it is shown in figure 5. These new services can be considered as black-boxes that interact with the environment through their ports.

Definition 4: Orchestrators The behavior of an orchestrator Σ is defined by the parallel composition of its lower level timed web services $\sigma_i, ..., \sigma_n$, by hiding all their internal actions. Formally,

$$\Sigma = \|_{i=1..n} \sigma_i \setminus INT$$

For example, the specification of the orchestrator that results from the composition of both services TV and WC services is like follows:

$$Orchestrator = (TV \parallel WC) \setminus \{Code\_Generation, Code\}$$

[1] In Timed CSP, the ? symbol represents "a waiting for requests" operation and the ! symbol represents a "send request" operation.

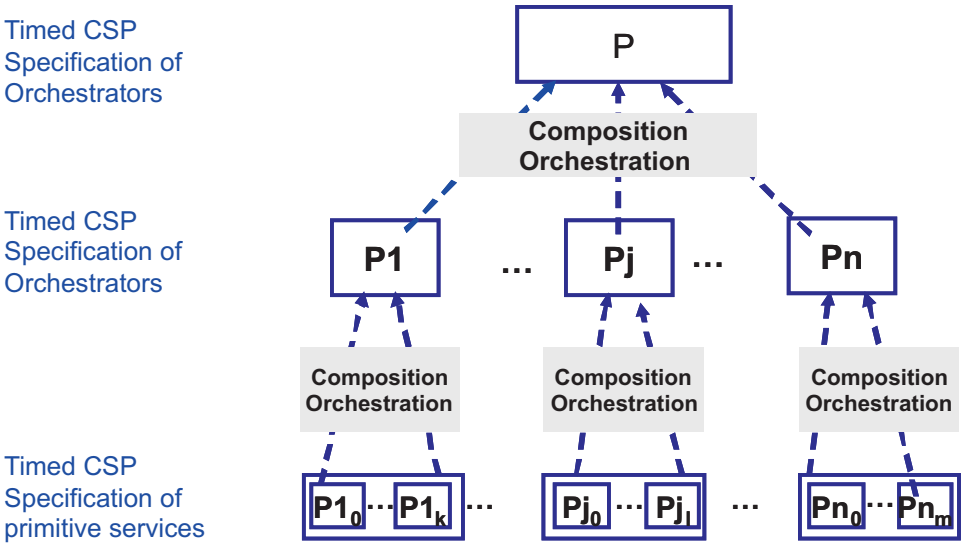

Fig. 5. Bottom-up service orchestration

6 Mapping from Timed CSP to WS-BPEL

In this section we establish the transformation rules between Timed CSP and WS-BPEL XML format. Table 1 summarizes these rules.

Events to basic activities The main elements of Timed CSP framework are events. *Events* represent atomic indivisible actions in the behavior of a process and, thus, correspond to messages in WS-BPEL. These events may be either reception or sending actions.

- The reception actions correspond to the basic activity $< receive > .. < /receive >$.
- The sending actions may correspond to $< reply > .. < /reply >$ or to $< invoke > .. < /invoke >$ basic activities, depending on whether the sending action is an output-response to a previous request or a call to another service with input parameters, respectively.

Processes to structured activities

- $SKIP$ is a process that indicates a successful termination and can be mapped to the *terminate* activity.
- $STOP$ is a deadlock stable process that does not engage in any external communication. This process models abnormal activity termination and, thus, it corresponds to *terminate* activity.

Timed CSP	WS-BPEL
SKIP	empty
STOP	terminate
$P_1; P_2; \ldots; P_n$	<sequence> Activity P_1 .... Activity P_n <\sequence>
$P_1 \| \| P_2 \| \| \ldots \| \| P_n$	<flow> Activity P_1 .... Activity P_n <\flow>
$P \overset{t}{\triangleright} Q$	<pick> <Onmessage.....\> <OnAlarm...\> <\pick>
Wait t;	Wait for= "0 " until="t"
a ? P	< sequence > < invoke, receive or reply.../ > activityP < /sequence >

Table 1. Mapping from Timed CSP to WS BPEL

– the sequence $P_1; P_2; ..P_n$ obviously, corresponds to

$$\begin{array}{c} < sequence > \\ P_1 \\ P_2 \\ .. \\ P_n \\ < /sequence > \end{array}$$

– $a \rightarrow P$ is the process that is initially prepared to engage in an event a and, then, it behaves like P. This process corresponds to:

$$\begin{array}{l} < sequence > \\ \qquad < invoke, receive\ \ or\ \ reply.../ > \\ \qquad\qquad activityP \\ < /sequence > \end{array}$$

- $P1||P2$ corresponds to a parallel execution of the structured activities $P_1, P_2.., P_n$, expressed as $< flow > P_1, P_2, P_n < /flow >$.
- Delay *wait t* corresponds to:

$$\begin{array}{c} < wait > \\ < for > t < /for > \\ < /wait > \end{array}$$

- A time-sensitive choice $(a \rightarrow P \overset{t}{\triangleright} Q$ is a process that initially gives the turning on priority to the process $(a \rightarrow P)$. If the event a does not occur in t units of time, a timeout occurs and the process behaves as Q and P never turns itself on. This choice corresponds to a "pick" activity with an $< onMessage >$ activity and an $< onAlarm >$ one.

$$\begin{array}{l} < pick > \\ \quad \begin{array}{c} < onMessage > .. < /onMessage > \\ ActivityP \\ < onAlarm >< for > n < /for >< /onAlarm > \\ activityQ \end{array} \\ < pick > \end{array}$$

7 Conclusions and Future Work

Despite the fact that time in which processes and services have to take place is capital in system design, there exists few proposals that address the capture of time constraints when composing services. In this paper we have presented a set of techniques for the orchestration and choreography of timed web services that combines UML-RT, Timed CSP and WS-BPEL. This approach enables the verification of some of the properties of real-time services while using a user friendly notation.

The techniques presented above will set the grounding for a model based approach intended to develop time-constrained business processes based on web services. This approach will be defined as a set of transformations between different models, as in a Model-Driven Engineering (MDE) methodology.

The intended development process can be summarized as follows:

1. **CIM level.** The business processes are designed using timed activity diagrams as presented in previous work [1].
2. **PIM level.** The specification of services and its composition is carried out by UML diagrams and its corresponding Timed CSP and WS-BPEL processes.
3. **PSM level.** An implementation of the models is specified in programming languages like RT-CORBA, Java-RT, etc..

References

1. Benghazi, K., Garrido, J.L., Noguera, M., Hurtado, M.V., Chung, L.: Extending and formalizing uml 2.0 activity diagrams for the specification of time-constrained business processes. In: RCIS. (2010)
2. Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: ICWS '04: Proceedings of the IEEE International Conference on Web Services, Washington, DC, USA, IEEE Computer Society (2004) 43
3. Schneider, S.: Concurrent and Real–Time Systems – The CSP Approach. John Wiley & Sons, Ltd., Chichester, England (2000)
4. Schneider, S.: An operational semantics for timed csp. Inf. Comput. **116**(2) (1995) 193–213
5. Hao, P, J.S.D.J.S., Zhang, X.: Reasoning about timed csp models. In: 14th International Symposium on Formal Methods (FM'06). (2006)
6. Roscoe, A.W., Hoare, C.A.R., Bird, R.: The Theory and Practice of Concurrency. Prentice Hall PTR, Upper Saddle River, NJ, USA (1997)
7. Benghazi, K.: Medistam-RT: Metodología de Diseño y Análisis de Sistemas de Tiempo-Real. Phd Thesis, University of Granada, Spain (2009)
8. Rauf, I., Iqbal, M.Z.Z., Malik, Z.I.: Uml based modeling of web service composition - a survey. In: SERA '08: Proceedings of the 2008 Sixth International Conference on Software Engineering Research, Management and Applications, Washington, DC, USA, IEEE Computer Society (2008) 301–307
9. Bauer, B., Müller, J.P.: Mda applied: From sequence diagrams to web service choreography. In: ICWE. (2004) 132–136
10. Cambronero, M.E., Diaz, G., Pardo, J.J., Valero, V.: Using uml diagrams to model real-time web services. In: ICIW '07: Proceedings of the Second International Conference on Internet and Web Applications and Services, Washington, DC, USA, IEEE Computer Society (2007) 24
11. Yeung, W.: Csp-based verification for web service orchestration and choreography. Simulation **83**(1) (2007) 65–74
12. Martens, A.: Analyzing web service based business processes. In: FASE. (2005) 19–33
13. Cámara, J., Canal, C., Cubo, J., Vallecillo, A.: Formalizing wsbpel business processes using process algebra. Electron. Notes Theor. Comput. Sci. **154**(1) (2006) 159–173
14. Puhlmann, F.: Soundness verification of business processes specified in the pi-calculus. In: OTM Conferences (1). (2007) 6–23
15. Hoare, C.A.R., Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM **21** (1985) 666–677
16. Dong, J.S., Zhang, X., Sun, J., Hao, P.: Reasoning about timed csp models (2004)
17. Dong, J.S., Hao, P., Sun, J., Zhang, X.: A Reasoning Method for Timed CSP Based on Constraint Solving. In: Formal Methods and Software Engineering. Springer Berlin / Heidelberg (November 2006) 342–359
18. Benghazi, K., Tuñón, M.I.C., Holgado, J.A., Mendoza, L.E.: Towards uml-rt behavioural consistency. In: ICEIS (3). (2007) 612–615
19. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. Technical report, OMG (2003)