Timed Modelling and Analysis in Web Service Compositions *

Raman Kazhamiakin
DIT, University of Trento
via Sommarive 14
38050, Trento, Italy
raman@dit.unitn.it

Paritosh Pandya
Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba
Mumbai 400 005, India
pandya@tifr.res.in

Marco Pistore
DIT, University of Trento
via Sommarive 14
38050, Trento, Italy
pistore@dit.unitn.it

Abstract

In this paper we present an approach for modelling and analyzing time-related properties of Web service compositions defined as a set of BPEL4WS processes. We introduce a formalism, called Web Service Timed State Transition Systems (WSTTS)*, to capture the timed behavior of the composite web services. We also exploit an interval temporal logic to express complex timed assumptions and requirements on the system's behavior. Building upon of this formalization, we provide techniques and tools for model checking BPEL4WS compositions against time-related requirements. We perform a preliminary experimental evaluation of our approach and tools with the help of the e-Government case study.*

1. Introduction

Web services provide the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols [9]. Service composition [11] is one of the most promising ideas underlying Web services: new functionalities can be defined and implemented by combining and interacting with pre-existing services. Different standards and languages have been proposed to develop Web service compositions. Business Process Execution Language for Web Services (BPEL for short) [2] is one of the emerging standards for describing a key aspect for the composition of Web services: the behavior of the service.

BPEL opens up the possibility of applying a range of formal techniques to the verification of the behavior of Web services, and different approaches have been defined for verifying BPEL [6, 12, 13, 7, 15]. We are interested in particular in those techniques that are applied to the verification of BPEL compositions: in this case, we have to verify the behaviors generated by the interactions of a set of BPEL processes, each specifying the workflow and the protocol of one of the services participating to the composition. Correctness of these compositions requires not only the satisfaction of qualitative requirements (e.g. deadlock freeness of the interaction protocols), but also of quantitative properties, such as time, performance, and resource consumption.

Time-related properties are particularly relevant in this setting. Indeed, in many scenarios we expect that a Web service composition satisfies some global timing constraints, and these constraints can be satisfied only if all the services participating to the composition are committed to respect their own local timing constraints. Consider for instance an e-government scenario, where the distributed business process requires the composition of information systems and functionalities provided by different departments or organizations (here, we will consider one of such scenarios, consisting in providing the authorization to open a site for the disposal of dangerous waste). The composite service can comply with the timing commitments with respect to the state regulations (e.g., the duration of document analysis phase) only if they are consistent with the time required by all participating actors to carry out their part of the process.

In this paper we present an approach for modelling and analyzing time properties of Web service compositions defined by a set of BPEL processes. We want to stress the fact that the time properties we want to model and analyze are those that are critical from the point of the business logic, i.e., they refer to the time required by the participating actor to carry out their tasks and take their decisions, and to the assumptions and constraints on these times that guarantee a successful execution of the distributed business processes. In e-government scenarios, their times are measured in hours and in days. The "technical" times, which are required, for instance, by the communications among BPEL processes and by the BPEL engines to manage incoming and outgoing message, are orders of magnitude smaller

*This work is partially funded by the MIUR-FIRB project RBNE0195K5, "Knowledge Level Automated Software Engineering", and by the MIUR-PRIN 2004 project "Advanced Artificial Intelligence Systems for Web Services".

(seconds if not milliseconds) and can be neglected in the scenarios above.

This work is based on previous results on the untimed verification of BPEL compositions [10]. In that framework, implemented as a part of the toolkit within the Astro project (`http://www.astroproject.org`), the BPEL processes are encoded as State Transitions Systems (STS), and then their composition is verified using NuSMV model checker. In this paper, we define the formalism of *Web Services Timed Transition Systems (WSTTS)*, which extends STS to allow for modelling the timed behavior of a BPEL composition. WSTTS are closely related to timed automata but incorporate design decisions and features consistent with the Web service composition. We also demonstrate how the duration calculus logic can be applied for the modelling of complex timed requirements in the domain. We provide techniques and tools for model checking the WSTTS specifications corresponding to such compositions. These techniques address basic verification tasks such as detecting the possibility of deadlock and of reaching error states under specific timing constraints. We also adapt Quantified Discrete-time Duration Calculus (QDDC) to perform a verification of the models enriched with complex duration constraints and requirements on top of the WSTTS Model [14]. The verification is carried out by first reducing the WSTTS models and the QDDC formulae to finite state automata, using techniques investigated in [16], and then by encoding these automata in the language of the NuSMV [5] model checker. The latter is then invoked to verify the desired property.

The structure of the paper is as follows. In Sect. 2 we introduce the e-government case study that describes the problem of analysis of time-related properties. Section 3 explains the formalism of WSTTS model, how the time-related properties are modelled in this formalism, and introduces the DC logic. The implementation of the timed analysis approach is discussed in Sect. 4 together with some experimental results on the presented case study. Conclusions and future work are presented in Sect. 5.

2. Case Study: e-Government Application

We illustrate our approach with the help of the real e-government application. The goal of the application is to provide a service that manages user requests to open sites for the disposal of dangerous waste. According to the existing Italian laws, such a request involves the interaction of different offices of the public administration, namely a Citizen Service, a Waste Management Office, a Secretary Service, a Procedure Manager, a Technical Committee, and a Political Board. In this application, the whole procedure is implemented as a composition of Web services that serve as interfaces to the processes of the above offices. We model the composition using BPEL [2] specifications to describe the interactions of the partners.

The high-level choreography model of the request management process is presented in Fig. 1. The procedure describes different phases of the application management. Briefly it can be summarized as follows. The request of the user is sent to the Waste Management Office (WMO) service that registers the protocol for the application. The registration time is used as a reference time point along the whole procedure. The set of provided application documents are sent to the Secretary Service. The latter should verify the documents integrity, and sends a notification to the user and the Procedure Manager (PM). If an additional documentation is needed, the user sends it to PM, which process them and initiate the main part of the procedure. At this point Technical Committee (TC) is invoked for the evaluation regarding the technical feasibility and ecological impact of the site. The reviews are collected by PM and a public conference is scheduled. Between two conference calls the citizen is allowed to modify the date. On the conference completion PM collects the evaluations and sends conference acts to the Political Board that is responsible to take a final decision on the application and provide it to the citizen.

Apart from the functional requirements, the execution of the process in the choreography should respect a set of timing requirements and constraints, dictated by Italian laws or by the agreement among the involved parties. These requirements include, for example, the following constraints (represented as callouts in Fig. 1):

- The period of time between the application registration and the notification of the Procedure Manager should not exceed 30 days;
- The first conference call should be made within 30 days after the registration and at least 10 days before the conference;
- Participants can change the date within 5 days after the preliminary call;
- The acts should be collected and sent to the Political Board service not later then 90 days after the conference announcement.

The behavior of the composition and the possibility to satisfy the above requirements depend on the time needed for the execution of the activities the involved parties are responsible for. Indeed, if the user does not provide additional documents within certain time, or if the technical analysis of the application lasts too long, the process may not terminate correctly. Moreover, it might be the case that the requirements and the composition are not consistent at all, i.e. there is no any possibility to execute the composition

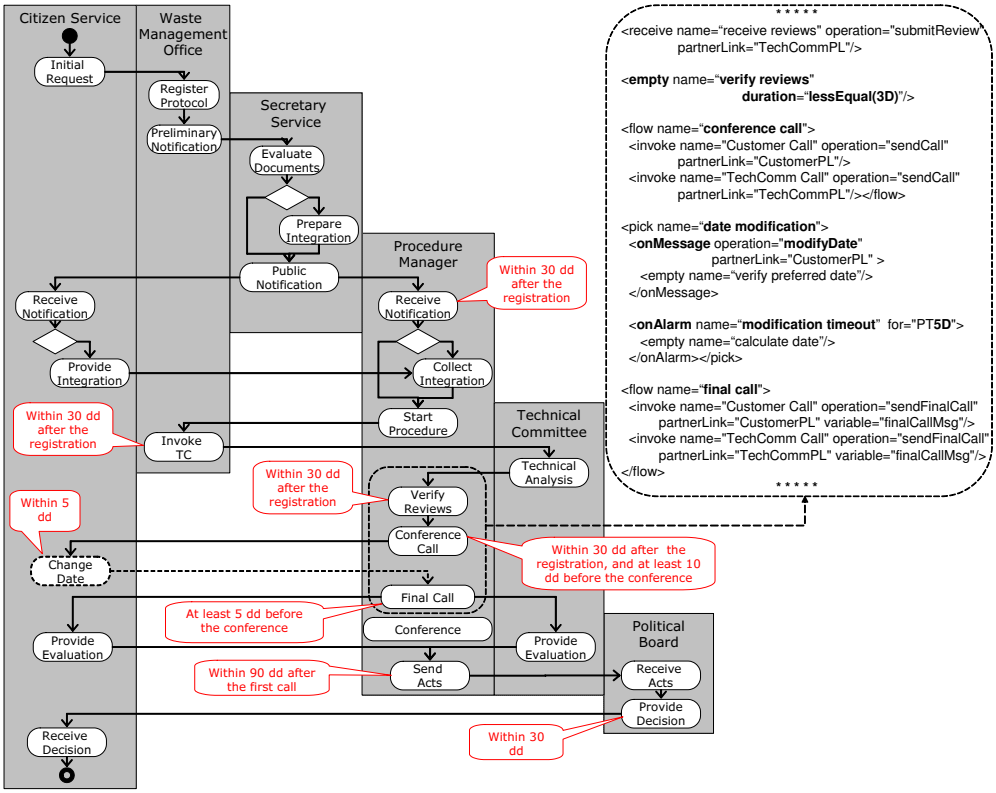

Figure 1. Waste management application processes

and satisfy the requirements. We remark that the critical parameters here do not refer to the communication time, which can be neglected, but with the duration of certain internal operations of the participants.

The analysis of time-related properties (e.g. whether the procedure can be completed within 90 days) requires the definition of timeouts (e.g. the timeout of the Procedure Manager waiting for the data modification) as well as of the answer times of the participating actors (e.g. the time required by the user to provide an additional documentation). While timeouts can be represented in BPEL, answer times can not, and require specific way to be modelled. In our framework, we assume that the answer times are negligible by default, and that activities that have a non-negligible answer time are annotated in the BPEL specification with an extra "duration" attribute.

The excerpt of the annotated BPEL specification is represented in Fig. 1. Here the fact that the modification is possible only within 5 days after the call, is represented with the BPEL event handler "date modification". The `onAlarm` activity is triggered if the timeout of 5 days passes and the user does not invoke the "modifyDate" operation. On the contrary, the activity "verify reviews" is equipped with a duration annotation that constraints its duration to be less than or equal to 5 days.

Besides duration annotations there is a need to model other time assumptions and constraints that can not be represented with durations associated to the activities and require more powerful notations. For instance, consider the requirement that the interval from the registration to the conference call should not exceed 30 days, and it is followed by the interval of length of at least 10 days, ending with the conference. In order to be able to handle these properties, it is necessary to provide models of the behaviors of BPEL Web services that allow for an explicit representation of time. Moreover, it is necessary to exploit and adapt the existing analysis techniques for reasoning about time to our problem domain. In the following sections we demonstrate how these issues can be addressed with the help of the WSTTS model, the duration annotations and (subclass of) duration calculus for complex time requirements.

3. Web Service Timed Transition System

The behavior of the Web service is described by sequences of activities. The semantic of these activities and the execution time depend on their type. For instance, the `onAlarm` activity is fired immediately at the moment when the timeout expires. The assignment of variables may be considered as an instantaneous activity, while the service

invocation operation may require an arbitrary amount of time. In order to model such behavior, we propose the Web Service Timed Transition System (WSTTS) model, which adopts the formalism of *timed automata* for capturing the aspects specific to the Web service domain. In this formalism, the fact that the operation takes certain amount of time is represented by time increment in the state, followed by the immediate execution of the operation. In order to guarantee that the transition will take place at the right moment of time, the states and transitions of timed automata are annotated with the invariants and guards of the special clock variables.

Intuitively, WSTTS is a finite-state machine equipped with set of clock variables[1]. The values of these clock variables increase with the passing of time. A Web service composition thus is represented as a network of several such automata, where all clocks progress synchronously. In this model, the states of the WSTTS are equipped with the *state invariants* that express simple conditions over clocks. The state invariants should be true when the system is in the state. Analogously, transitions are annotated with the set of *guards* and *resets*. The former represent simple conditions over clocks, and the latter are used to reset values of certain clocks to zero. The semantic of WSTTS is defined as a labelled transition system, where either the time passes or a transition from one state to another immediately takes place.

Let X be a set of clocks. The constraints on the clock values $\Phi(X)$ are of the form $true \mid x \sim c \mid \phi_1 \wedge \phi_2$, where $\sim \in \{\leq, <, =, \neq, \geq, >\}$, $x \in X$, and $c \in \mathbb{T}$, a domain of time values.

Definition 1 (WSTTS)
WSTTS is a tuple (S, s_0, A, Tr, Inv)*, where*

- S *is the set of states and* s_0 *is the initial state;*
- A *is a set of actions. Each action is either an input action* $?m$*, an output action* $!m$ *or an internal* τ *action;*
- $Tr \subseteq S \times A \times \Phi \times 2^X \times S$ *is the set of transitions with an action, a guard, and a set of clocks to be reset;*
- $Inv : S \to \Phi(X)$ *assigns invariants to the states.*

In the definition, the effect of the transition from the state s to s' is to perform a communication or an internal action $a \in A$, and to reset set of timers to zero. The transition is possible only if the guard condition evaluates to true in the source state.

Now we define the semantics of a WSTTS. A clock valuation is a function $u : X \to \mathbb{T}$ from the set of clocks to the domain of time values. Let $\mathbb{T}_C$ denote a set of all clock valuations. Let $u_0(x) = 0$ for all $x \in X$. We will write $u \in Inv(s)$ to denote that u satisfy $Inv(s)$.

Definition 2 (Semantics of WSTTS)
Let (V, S, s_0, A, Tr, Inv) *be a WSTTS. The semantics is defined as a labelled transition system* $(\Gamma, \gamma_0, \to)$*, where* $\Gamma \subseteq S \times \mathbb{T}_C$ *is a configuration,* $\gamma_0 = (s_0, u_0)$ *is an initial configuration, and* $\to \subseteq \Gamma \times \{A \cup tick\} \times \Gamma$ *is a transition relation such that:*

- $(s, u) \xrightarrow{tick} (s, u + d)$*, if* $(u + d) \in Inv(s)$*, and*
- $(s, u) \xrightarrow{a} (s', u')$*, if there exists* $(s, a, \phi, Y, s') \in Tr$*, such that* $u \in \phi$*,* $u' = u[Y \mapsto 0]$*, and* $u' \in Inv(s')$*.*

That is, either the system remains in the same state and time passes, or a certain transition immediately takes place.

We define a Web service composition as a *WSTTS network*. The WSTTS network $PP = (X, P_1 \parallel \cdots \parallel P_n)$ consists of n WSTTS P_i over common set of clocks X. The semantics of the WSTTS network is given in terms of global timed transition system (GTTS). We use $\bar{s} = (s_1, \ldots, s_n)$ to denote a state vector, $\bar{s}_0$ to denote an initial state vector, and $\bar{s}[s_i/s_i']$ to denote a state vector, where the element s_i is replaced by s_i'.

Definition 3 (GTTS)
Let $(X, P_1 \parallel \cdots \parallel P_n)$ *be a WSTTS network.* Global timed transition system *has the form* $(\Gamma, \gamma_0, \to)$*, where* $\Gamma \subseteq \langle S_1 \times \cdots \times S_n \rangle \times \mathbb{T}_C$*,* $\gamma_0 = (\langle s_{01}, \ldots, s_{0n} \rangle, u_0)$*, and* $\to \subseteq \Gamma \times \{A \cup tick\} \times \Gamma$ *is a global transition relation defined by:*

- $(\bar{s}, u) \xrightarrow{tick} (\bar{s}, u + d)$*, if* $(u + d) \in \wedge_i Inv_i(s_i)$*;*
- $(\bar{s}, u) \xrightarrow{\tau} (\bar{s}[s_i/s_i'], u')$*, if there exists a transition* $(s_i, \tau, g, Y, s_i') \in Tr_i$*, s.t.* $u \in g$*,* $u' = u[Y \mapsto 0]$*, and* $u' \in \wedge_i Inv_i(s_i)$*;*
- $(\bar{s}, u) \xrightarrow{m} (\bar{s}[s_i/s_i', s_j/s_j'], u')$*, if there exist a transition* $(s_i, ?m, g_i, Y_i, s_i') \in Tr_i$*, and a transition* $(s_j, !m, g_j, Y_j, s_j') \in Tr_j$*, s.t.* $u \in g_i \wedge g_j$*,* $u' = u[Y_i \cup Y_j \mapsto 0]$*, and* $u' \in \wedge_i Inv(s_i)$*.*

In other words, the model of GTTS allows for three kinds of transitions: a time passing transition, where all clocks increment their values; an internal transition of a particular WSTTS, and a shared communication action of two WSTTS.

3.1. Mapping BPEL constructs to WSTTS

We now give the definition of BPEL constructs in terms of the WSTTS formalism. We remark that, by default, all the activities of the underlying BPEL process are modelled as instantaneous. The fact that a particular activity may have

[1] It is also equipped with the set of non-timed variables of finite domains. For the sake of simplicity, we omit them in the formalism.

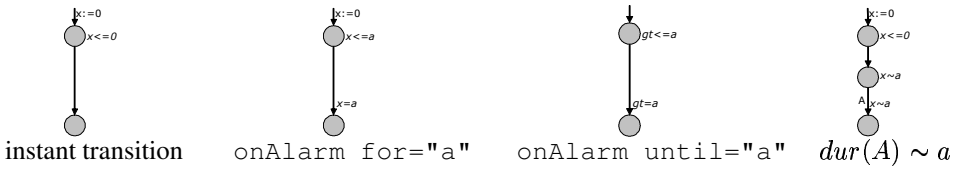

Figure 2. Time-related constructs as WSTTS

certain duration is expressed explicitly through *duration annotations* that allow to specify bounds of the activity duration[2].

In this way, until explicitly specified, all internal and message output activities are modelled as instant. Such a transition is semantically equivalent to adding an extra clock `x` to the source state of the transition, and the invariant of the state is `x<=0`. Hence, time can not pass in the source state of the instant transition. On the contrary, input activities do not require such addition, since they are blocked until corresponding output takes place, and therefore time can pass.

BPEL also defines activities that explicitly reference time. In particular, the `onAlarm` activity is fired used to represent timeouts and is modelled as an event handler. This activity has two forms. In the first form it is fired when certain time has passed. In the BPEL code represented in Fig. 1 this activity is used in event handler to set timeout to 5 days for the conference date modification. In the second form, `onAlarm` is fired if the current absolute time has the specified value[3]. In order to model the absolute time referenced in this activity, a special clock is added to the WSTTS network model, namely *global_timer*. It can be explicitly set to a certain value in the beginning of the execution, and is never reset later.

As we mentioned above, it is possible to specify that certain activity is not instant. In this way one is allowed to express time-related assumptions on the process execution, like service response time, duration of some internal operation, etc. In this case the activity is explicitly annotated with the duration constraints. This annotation is used in our case study, for example, to denote that the duration of the "verify activity" is less than or equal to 5 days (Fig. 1). Such constraints are conjunctions of the clauses of the form $dur(A) \sim c$, where $\sim \in \{<, >, \leq, \geq, =\}$. In the WSTTS formalism this annotation is semantically equal to the sequence of two transitions. First transition is instant and it resets the clock `x`. The second transition has the guard that evaluates to true, if the value of the clock `x` satisfies the duration constraints. Analogous constraint is defined in the intermediate state.

Figure 2 represent the translation of these constructs.

3.2. Specifying Time Requirements

While the constructs described above enable the explicit coding of simple time-related properties of Web service compositions, we often encounter complex timing assumptions and requirements which are hard to state as timeout or a constraint of activity duration. Such requirements may express the time intervals between events (or a sequences of events), time bounds on some condition to hold or even complex logical combinations on them.

In order to express such properties we exploit a subset of duration calculus (DC) [4]. It allows us to express properties of finite sequences of behaviors and to measure the duration of a given behavioral fragment.

Let P range over propositional variables, D, D_1, D_2 over DC formulae, c range over natural number constants, and $\sim \in \{<, \leq, =, >, \geq, \}$. The syntax of the DC formula is:

$$D := \lceil P \rceil^0 \mid \lceil\lceil P \rceil \mid D_1 \frown D_2 \mid D_1 \wedge D_2 \mid \neg D \mid len \sim c$$

DC formulas are evaluated over finite behaviors, i.e., over finite sequences of valuations of propositional variables $VAL(Pvar)^*$.

The constructs above have the following intuitive meaning:

- $\lceil P \rceil^0$ holds for the behavior consisting of a single state satisfying P;
- $\lceil\lceil P \rceil$ requires P to hold at all the states of the behavior (except the last state);
- $D_1 \frown D_2$ splits the behavior into two subintervals, such that D_1 holds for the first subinterval, and D_2 holds for the second one;
- $D_1 \wedge D_2$ requires both formulas to hold, while $\neg D$ requires that D is not satisfied on the behavior;
- $len \sim c$ relates the duration of the interval with the constant value c (i.e. greater, equal, etc.).

[2] We stress once more that our goal is to analyze the time properties that are critical for the business logic, and neglect the smaller "technical" times due, e.g. the communications.

[3] Another BPEL activity that deals with time is the activity `wait`. This activity is blocked for certain time period and is translated into WSTTS analogously

Additionally, we write $\Diamond D = true \frown D \frown true$, if D holds for some subinterval of the behavior; $\Box D = \neg \Diamond \neg D$ denotes, that D holds for all subintervals.

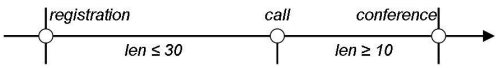

Figure 3. DC requirement example

Consider, for instance, the requirement that the interval from the protocol registration to the conference call should not exceed 30 days, and that from the call to the conference at least 10 days should pass. The requirement may be graphical represented as shown in Fig. 3 and expressed with the following formula:

$$\Box(\lceil registration \rceil^0 \frown true \frown \lceil conference \rceil^0 \rightarrow (len \ \leq 30) \frown \lceil call \rceil^0 \frown (len \ \geq 10))$$

The formula says that for all intervals of the behavior, if the registration happens at the beginning of the interval and the conference at the end, then the interval consists of two intervals with the call in between, such that the duration of the first is less than 30 days, and the duration of the second does not exceed 10 days.

4. Implementation of Timed Analysis

We have implemented the ideas presented above as a prototype tool that allows for the timed analysis of Web service compositions. The tool inputs the initial composition of BPEL processes, enriched with the duration constructs, together with complex properties and translates it into the specification suitable for the formal techniques, such as model checking. In this implementation we adopt discrete model of time, and use (subset of) Quantified Discrete-time Duration Calculus [14] to express complex time requirements under this model. The analysis based on the dense model of time under certain conditions may be implemented in a similar way.

The tool performs the transformation of the composition into the finite state automata representation and reflect the operational semantics of the global TTS given above. The clock variables are represented as global integer variable that synchronously increment their values when the time elapse transition happens. A special $tick$ variable is used to denote this event. The results of [1] ensure the finiteness of the resulting specification. In particular, it is shown that in the discrete model of time the clock variables may be bounded without affecting the behavior of the system.

The complex timed requirements may be used in the analysis of the composition in the following ways. First, the composition specification M can be directly verified against a property represented with the QDDC formula D. For this purposes we construct an finite state automaton A that recognizes all and only the behaviors that satisfy the formula $\neg D$ ([14]). If the synchronous (i.e. lock-step) product $(M \times A(\neg D))$ of the specification and the automaton for the property negation is not empty, then the behavior of the composition violates the property D.

Second, the complex time properties may express assumptions or constraints on the system behavior. In this case the tool builds a product $(M \times (A(D_1) \times \cdots \times A(D_n)))$ of the specification and the properties automata. Indeed, this product describes the specification model such that its behavior satisfies all the constraints specified. One can use this product for further analysis of the composition (e.g. for CTL or LTL model checking).

In order to illustrate the approach represented in the paper we have conducted a set of experiments on the analysis of the presented case study in different settings. In particular, in one set of experiments we assigned different bounds on the activity durations, and checked the composition for the deadlocks. Another set of experiments dealt with complex requirements expressing the bounds on intervals between various events (e.g. between application registration and conference call). The requirements were verified directly, or were used to model behavioral constraints. We have used the NuSMV model checker for verifying the corresponding finite state automata model. In the experiments the state space ranges from 10^{11} to 2×10^{12} states, and the verification time ranges from 0.6 to 15 seconds. Whenever the property is violated, the model checker generates a counterexample that represents an execution demonstrating the violation (e.g. a trace where both registration and conference call happened, but the time between these events was greater then the required bound).

5. Conclusions

We presented an approach for modelling and analyzing of time-related properties on Web service compositions defined by a set of BPEL processes. This approach is based on the formal model, Web Service Timed Transition System, that allows to take into account timed behavior of such compositions. We demonstrated how BPEL time-related constructs can be expressed in this formalism. Moreover, we presented a way to express various time-related requirements and assumptions using both simple modelling constructs or complex DC formulas particularly suitable for expressing such properties. The presented approach enables verification of Web service compositions against time properties using model checking techniques.

The problem of the Web service compositions analysis, in particular of BPEL processes, is investigated in the

works of [6, 12, 13, 7, 15]. While providing facilities for the verification of processes or their compositions, these approaches do not take time-related properties of composition behaviors into account. The work that is closer to ours is presented in [8]. In this work, a formal model of BPEL processes, μ-BPEL, is presented which allows for mapping from this formalism to a network of timed automata. However, [8] does not provide a way to explicitly specify the transition or state duration bounds, or complex time-related assumptions and requirements as those we model with DC formulas. In [3] temporal abstractions are exploited for the compatibility and replaceability analysis of Web service protocol. In that model one can specify when certain transitions must or may happen, similarly to what we achieve with our duration annotations. However, [3] does not address the problem of the verification of these time properties. Moreover, their temporal abstraction are simple with respect to the set of properties we can express in our approach.

There are several direction for further research. In particular, we are working on the optimizations of the translations from BPEL to NuSMV code and applications of better analysis techniques that give a possibility to drastically improve the verification performance. Another line of research is to replace NuSMV with a model checker, such as UPPAAL, that can verify timed properties without requiring the generation of FSA (even if the results of [16] show that this does not necessary leads to a better performance in the verification).

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994.

[2] T. Andrews, F. Curbera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weeravarana. Business Process Execution Language for Web Services (version 1.1), 2003.

[3] B. Benatallah, F. Casati, J. Ponge, and F. Toumani. On Temporal Abstractions of Web Service Protocols. In *Procs of CAiSE Forum*, 2005.

[4] Z. Chaochen, C.Hoare, and A. Ravn. A Calculus of Durations. In *IPL*, 1991.

[5] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(4), 2000.

[6] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based verification of web service compositions. In *Proc. of the 18^{th} International Conference on Automated Software Engineering (ASE'03)*, 2003.

[7] X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proc. WWW'04*, 2004.

[8] P. Geguang, Z. Xiangpeng, W. Shuling, and Q. Zongyan. Towards the Semantics and Verification of BPEL4WS. In *Proc. WS-FM'04, ENTCS*, 2004.

[9] S. Graham, S. Simenov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*. Sams, 2001.

[10] R. Kazhamiakin and M. Pistore. Parametric Communication Model for the Verification of BPEL4WS Compositions. In *Proc. WS-FM'05*, 2005. To appear.

[11] R. Khalaf, N. Mukhi, and S. Weeravarana. Service Oriented Composition in BPEL4WS. In *Proc. WWW2004*, 2004.

[12] S. Nakajima. Model-checking verification for reliable web service. In *Proc. OOPSLA'02 Workshop on OOWS*, 2002.

[13] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proc. WWW'02*, 2002.

[14] P. Pandya. Specifying and Deciding Qauntified Discrete-time Duration Calculus formulae using DCVALID. In *Proc. Real-Time Tools, RTTOOLS'2001*, 2001.

[15] M. Pistore, M. Roveri, and P. Busetta. Requirements-Driven Verification of Web Services. In *Proc. WS-FM'04, ENTCS*, 2004.

[16] D. Thomas, P. Pandya, and S. Chakraborty. Scheduling clusters in model checking of real-time systems. Technical Report TR-04-16, CFDVS, IIT Bombay, September 2004.