Modelling and Analysis of Time-related Properties in Web Service Compositions

Raman Kazhamiakin[1], Paritosh Pandya[2], and Marco Pistore[1]

[1] DIT, University of Trento
via Sommarive 14, 38050, Trento, Italy
{raman,pistore}@dit.unitn.it
[2] Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba, Mumbai 400 005, India
pandya@tifr.res.in

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 on this formalization, we provide tools and techniques 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 [1]. Service composition [2] 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) [3] 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 [4–7, 13]. 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), 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 composition satisfies some global timing constraints, which can be satisfied only if all the participating services 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. The composite service can comply with the timing commitments w.r.t. the state regulations (e.g., the duration of document analysis phase) only if they are consistent with the time required by the participants to carry out their part of the process.

In this paper we present an approach for modelling and analyzing time properties of WS 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 business logic point, i.e., they refer to the time required by the participants 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 composition scenario. 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 (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 [8]. In that framework, implemented as a part of the Astro project toolkit (`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)*, extending STS to allow for modelling the timed behavior of a 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, and adapt Quantified Discrete-time Duration Calculus (QDDC) [9] to perform a verification of WSTTS models under these requirements. The verification is carried out by first reducing the WSTTS models and the QDDC formulae to finite state automata [9], and then by encoding these automata in the language of the NuSMV [10] model checker. The latter is then invoked to verify the desired property.

The paper is structured as follows. In Sect. 2 we introduce the e-government case study that describes the problem of analysis of time-related properties. The formalism of WSTTS model, the representation of time-related properties, and DC logic are discussed in Sect. 3. Section 4 discusses the implementation of the analysis approach and experimental results, and Sect. 5 presents conclusions.

2 Case Study: e-Government Application

We illustrate our approach with the real e-government application. The goal of the application is to provide a service that manages user requests to open sites

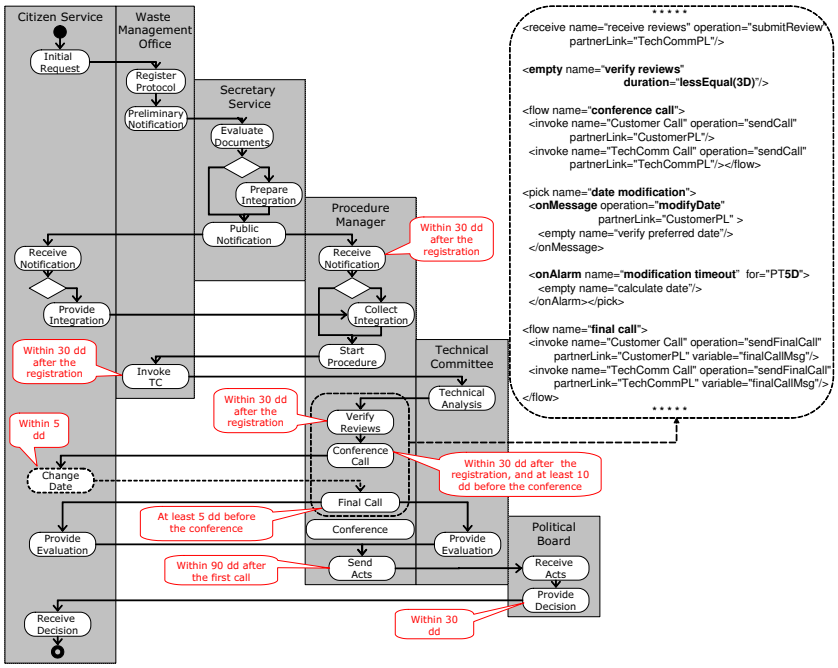

Fig. 1. Waste management application processes

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 (WMO), 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 specifications to describe partners interactions. The high-level choreography model of the request management procedure is presented in Fig. 1. The procedure describes different phases of the application management where the request is registered, the documentation is evaluated and collected, tha application is analyzed regarding the ecological impact of the site, the public conference is scheduled and organized, and final decision is provided.

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 (callouts in Fig. 1) specify, for example, that the period of time between the application registration and the notification of the `Procedure Manager` should not exceed 30 days, or that the participants can change the date within 5 days after the preliminary call.

The behavior of the composition and the possibility to satisfy these requirements depend on the time needed for the execution of the activities the involved parties are responsible for. The critical parameters here refer to the duration of

internal activities of the participants, and not to the communication time, which can be neglected.

The analysis of time-related aspects of the compositions requires explicit representation of timeouts, operation durations, and even complex properties expressing various timing requirements. While timeouts can be represented in BPEL, durations and timing requirements 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 duration are annotated in the BPEL specification with an extra "duration" attribute. In Fig. 1 an excerpt of the annotated BPEL is represented. Here a BPEL event handler "date modification" is used to model 5-days bound for the conference data change. That is, the `onAlarm` activity is triggered if the user does not invoke the "modifyDate" operation within 5 days. On the contrary, the internal activity "verify reviews" is equipped with a duration annotation to express that certain time may be used for the reviews analysis. Timing requirements, however, can not be represented with durations associated to the activities and require more powerful notations. Consider, for instance, 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 such aspects, it is necessary to provide model of the BPEL process behavior that allows 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 duration calculus for complex time requirements.

3 Web Service Timed Transition System Model

In order to model the behavior of the BPEL process compositions, 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[3]. 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 and should be true when the system is in the state. Analogously, transitions are annotated with the set of *guards* that represent simple conditions

[3] 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.

over clocks, and *resets* that are used to reset values of certain clocks to zero. The semantic of WSTTS is defined as a 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 input $?m$, output $!m$ or internal τ actions;*
- *$Tr \subseteq S \times A \times \Phi \times 2^X \times S$ 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 evaluates to true in the source state.

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 clock valuations. Let $u_0(x) = 0$ for all $x \in X$. We write $u \in Inv(s)$ to denote that u satisfies $Inv(s)$.

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 exists a transition $(s, a, \phi, Y, s') \in Tr$, such that $u \in \phi$, $u' = u[Y \mapsto 0]$, and $u' \in Inv(s')$.*

We define a Web service composition as a *WSTTS network*. The WSTTS network consists of n WSTTS P_i over common set of clocks X. The semantics of the 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:*

- *$(\bar{s}, u) \xrightarrow{tick} (\bar{s}, u + d)$, 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 $(s_i, \tau, g, Y, s'_i) \in Tr_i$, 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 exists $(s_i, ?m, g_i, Y_i, s'_i) \in Tr_i$ and $(s_j, !m, g_j, Y_j, s'_j) \in Tr_j$, 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 transition of GTTS is either a time passing transition, an internal transition of a particular WSTTS, or a shared communication action of two WSTTS.

Fig. 2. Time-related constructs as WSTTS

Mapping BPEL constructs to WSTTS. We now give the definition of BPEL constructs in terms of the WSTTS formalism (see Fig. 2). 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 certain duration is expressed explicitly through *duration annotations* that allow to specify bounds of the activity duration[4].

In this way 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. Input activities do not require such an 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 activity `onAlarm` is used to represent timeouts and is modelled as an event handler. This activity has two forms: in the first it is fired when certain time passes (in Fig. 1 it is used in event handler to set up a 5 days timeout for the date modification); and in the second it is fired if the current absolute time has the specified value[5]. We model this absolute time using a special clock added to the WSTTS network model, namely *global_timer*. It is 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 certain duration for the activity. In this case the activity is explicitly annotated with the duration constraints (e.g. duration of activity "verify reviews" in 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 and the intermediate state have the guards that evaluate to true, if the value of the clock `x` satisfies the duration constraints.

Specifying Time Requirements. While the constructs described above enable the explicit coding of simple time-related properties of WS compositions, we often encounter complex timing requirements which are hard to model with above constructs. 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.

[4] 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.

[5] 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

In order to express such properties we exploit a subset of duration calculus (DC) [11]. 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 over natural number constants, and $\sim \in \{<, \leq, =, >, \geq, \}$. The DC formula syntax 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 but the last states of the behavior;
- $D_1 \frown D_2$ 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.

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.

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, can be 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 every interval of the behavior that starts with the registration and ends with the conference consists of two subintervals with the call in between, such that the first lasts at most 30, and the second at least 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 [9] to express complex time requirements under this model. Under certain conditions the dense model of time may be analogously implemented.

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 [12] ensure that

for discrete time model the clock variables may be bounded without affecting the behavior of the system and therefore the resulting specification is finite.

The complex timing requirements are used in the composition analysis 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 $\neg D$ ([9]). 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. This product restricts the specification model to behaviors that satisfies all the specified constraints. 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 and presented a way to express time-related requirements using both simple modelling constructs or complex DC formulas particulary suitable for expressing such properties. The presented approach enables verification of WS compositions using model checking techniques.

The problem of the WS compositions analysis, in particular of BPEL processes, is investigated in the works of [4–7, 13]. 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 [14]. In this work, a formal model of BPEL processes, μ-BPEL, is presented that allows for mapping to a network of timed automata. However, [14] 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 [15] 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. The work does not address the problem of the verification of these time properties and the abstractions are simple with respect to the set of properties we can express in our approach.

There are several directions 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.

References

1. Graham, S., Simenov, S., Boubez, T., Daniels, G., Davis, D., Nakamura, Y., Neyama, R.: Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI. Sams (2001)
2. Khalaf, R., Mukhi, N., Weeravarana, S.: Service Oriented Composition in BPEL4WS. In: Proc. WWW'04. (2004)
3. Andrews, T., Curbera, F., Dolakia, H., Goland, J., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weeravarana, S.: Business Process Execution Language for Web Services (version 1.1) (2003)
4. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based verification of web service compositions. In: Proc. ASE'03. (2003)
5. Nakajima, S.: Model-checking verification for reliable web service. In: Proc. OOPSLA'02 Workshop on OOWS. (2002)
6. Narayanan, S., McIlraith, S.: Simulation, Verification and Automated Composition of Web Services. In: Proc. WWW'02. (2002)
7. Fu, X., Bultan, T., Su, J.: Analysis of Interacting BPEL Web Services. In: Proc. WWW'04. (2004)
8. Kazhamiakin, R., Pistore, M.: Parametric Communication Model for the Verification of BPEL4WS Compositions. In: Proc. WS-FM'05. (2005)
9. Pandya, P.: Specifying and Deciding Qauntified Discrete-time Duration Calculus formulae using DCVALID. In: Proc. RTTOOLS'01. (2001)
10. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NuSMV: a new symbolic model checker. Int. Journal on STTT (2000)
11. Chaochen, Z., C.Hoare, Ravn, A.: A Calculus of Durations. In: IPL. (1991)
12. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. (1994)
13. Pistore, M., Roveri, M., Busetta, P.: Requirements-Driven Verification of Web Services. In: Proc. WS-FM'04, ENTCS. (2004)
14. Geguang, P., Xiangpeng, Z., Shuling, W., Zongyan, Q.: Towards the Semantics and Verification of BPEL4WS. In: Proc. WS-FM'04, ENTCS. (2004)
15. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On Temporal Abstractions of Web Service Protocols. In: Procs of CAiSE Forum. (2005)