Specification and Verification of Timed Semantic web Services

Amel Boumaza
LIRE laboratory, Constantine 2 University
Constantine, Algeria
Spd_ing2006@yahoo.fr

Ramdane Maameri
LIRE laboratory, Constantine 2 University
Constantine, Algeria
ramdane.maamri@univ-constantine2.dz

Abstract – It is widely recognized that temporal aspects are indispensable for Web Service modeling. Unfortunately, the current Semantic web Services description languages suffer from the lack of useful concepts needed for timing description. For this purpose, we propose a global methodology for the specification of timing behavior with an extended OWL-S ontology and verification of temporal properties with UPPPAL tool. The applicability is illustrated through the multimodal transport use case.

Keywords –Web Services, OWL-s, Entry Sub-Ontology Of Time, Timed Automata, Uppaal Temporal Properties

1. INTRODUCTION

Automatic composition is the capability to automatically compose a set of services to fulfill user requirements when a single service is not available for fulfilling these requirements.
Composing services need an approach based on semantic descriptions, as the requester required functionally has to be expressed in a high-level and abstract way to enable reasoning procedures.
Efficiently, the semantic web community allows reasoning about web resources by explicitly declaring their preconditions and effects with terms defined precisely in ontologies.
Furthermore, in Semantic Web Services, the common need for temporal information is satisfied by the use of the temporal ontologies to allow using of temporal concepts. However, the complexity comes usually with the growing expressiveness; it becomes a challenging area to ensure correctness. Hence, the verification of Web Service flow becomes more and more important.
Formal methods are particularly well appropriate to address most of the aforementioned issues (e.g. composition and correctness). The majority of these are based on state-action models (e.g. labeled transition systems, timed automata, and Petri nets) or process models (e.g. π-calculus and other calculi) [1].

2. Related works

Recently, a diversity of concrete proposals from the formal methods community have emerged in order to verify the correctness of the web service composition which is based on state action models or process models [2].
In [3], a case study is presented that shows how descriptions of web services written in BPEL-WSCDL (Web Services Choregraphy Description Language) can be automatically translated to timed automata and subsequently be verified by Uppaal.
In [4] the authors provide an encoding of BPEL processes into web service timed state transition systems and discuss a framework in which timed assumptions expressed in the duration calculus [5] can be model checked.
In [6] a framework to automatically verify systems that are modeled in Orc[1] is proposed.

[1] Home page: http://orc.csres.utexas.edu/

Uppaal can be used to model check Orc models. The approach is demonstrated through a small case study.
In [7], the authors deal with the compatibility problem. The proposed framework allows applying model checker Uppaal to asynchronous Web services composition analysis by using a set of CTL formulas that characterize the different choreography compatibility classes defined and verifying deadlock free property.
In [8] was developed a method for verifying temporal consistency in the Web Service flow. Time ontology is added tothe OWL-S specification, and then the annotated OWL-S is transformed into formal model TCPN (Time Constraint Petri Net), then temporal consistency of Web Service flow is verified.
In [9], the authors use a WS-CDL (Web Services Choreography Description Language) description of composite Web services, and define an operational semantics for a translation of a subset of WS-CDL into a network of Timed Automata. Uppaal tool is used for the validation and verification of the generated timed automata.
It can be seen that there is little work on timing constraint satisfiability based on formal methods and which can combine semantic web approach. On the other hand and to our knowledge, there is no research conducted on bounded liveness property verification.
In this paper, we propose a global methodology for specifying and verifying timed semantic web services. First, we use OWL-S ontology to specify the web services. The Time-Entry ontology allows us to express temporal and timed concepts. The resulting model is transforming into a network of timed automata to be verified with Uppaal tool.

3. TIMED SEMANTIC WEB SERVICES SPECIFICATION

3.1. Owl-S Overview

OWL-S (Web Ontology Language for Services) is a high level ontology-based language for describing web service properties [10]. In this language, each web service is specified in three XML-based parts: service profile, which describes what the service does? Service model, which describes how does the service work (behave); and service grounding, which provides details on how to invoke a service through messages.OWL-S allows the description of the external behavior of a Web service by using a semantic model, in which each implicated atomic process is described semantically in terms of inputs, preconditions, outputs, and effects (IPOEs).
Composite processes are used to describe collections of processes (either atomic or composite) organized on the basis of following control flow structure:

- Sequence: A list of components to be done in order.
- If-then-else: the selection based on some obviously defined conditions.
- Choice: any of the given components may be chosen for execution.
- Repeat-while: the component is performed repeatedly while certain conditions are satisfied.
- Repeat-until: the component is performed repeatedly until some conditions hold.
- Any-Order: the components are performed in some unspecified order but not concurrently.
- Split: the components are activated and performed concurrently.
- Split+Join: the parallel execution is synchronizing with barrier synchronization.

3.2. Timing Constraint Specification

Based on Semantic Web, OWL-S describes the service semantic in different aspects. Unfortunately, it still lacks the definition of time constraints for Web Service which makes it difficult to verify temporal properties of the Web Service flow.
To provide support for describing temporal properties for OWL-S, we use an (entry) sub-ontology of time, which is much simpler than the full ontology of DAML-Time[2] which provides the basic temporal concepts and relations that most simple applications would need i.e. instants and, intervals [11].
By adding timing constraints to OWL-S, the time information related to the service can be defined easily. The entry sub-ontology provides quick access to the essential vocabulary in OWL for the basic temporal concepts and relations. It covers relations among instants and intervals and instant-like and interval-like events such as "before" and "overlaps". It includes measures for durations, so that we can say a course will last 1

[2]DAML-Time Homepage: http://www.cs.rochester.edu/~ferguson/daml/

hour and 30 minutes, and it also includes a clock and calendar terms so that we can say a course starts at 10:00am on Monday, December 17, 2013.
These basic temporal constraint relations are:

- Before (d_1, d_2): d_1 ends before d_2 starts and there is an interval between them.
- Meets (d_1, d_2): d_1 ends at the same instant when d_2 starts to execute.
- Finishes (d_1, d_2): d_2 starts after d_1 starts and before d_1 ends, and d_2 ends at the same instant with d_1.
- Overlaps (d_1, d_2): d_2 starts after d_1 starts and before d_1 ends, and d_2 ends after d_1 ends.
- Covers (d_1, d_2): d_1 execution interval includes d_1 execution interval.
- Starts (d_1, d_2): d_2 starts at the same instant with d_1, and d_2 ends after d_1 ends.
- Equals (d_1, d_2): d_2 starts and ends at the same instant with d_1.

Notice that, an interval d may be also written as [b, e] corresponding to beginning and ending point, respectively.

4. Timed Model Checking

Model Checking [12] is one of the most successful approaches for verifying temporal specifications of hardware and software systems. System properties are specified in temporal logic for which various formalisms exist. Classically two types of properties are distinguished, safety and liveness. In practical applications, safety properties are prevalent. Consequently, very efficient algorithms and tools have been developed for checking safety properties. A model-checking tool accepts system designs (called models) and properties (called specification) that models are expected to satisfy. The tool then outputs yes, if the given model satisfies given specifications and generates a counter example otherwise.

In this paper, we choose Timed Automata as underlying model, and TCTL [13] logic to specify the property to verify.

4.1. Timed automata

In this section we reply Timed Automata, which were introduced by Alur and Dill [14]. Timed Automata are extensions of finite state automata with constraints on timing behavior. The underlying finite state automata are augmented with a set of real time variables.

Definition1. A Timed Automaton A is a six-element tuple $(\Sigma, S, s_0, H, E, I)$ where:

- $\Sigma$is a finite set of actions,
- S is a finite set of locations,
- $s_0 \subseteq S$ is an initial location,
- H is a finite set of clocks,
- $E \subseteq S \times S \times \Sigma \times 2^H_{fn} \times \Phi(H)$ is a transition relation,
- $I$is a (location) invariant.

Each element e of E is denoted by $\langle s, s', a, \lambda, \varphi \rangle$ represents a transition from the location s to the location s', executing the action a, with the set $\lambda \subseteq H$ of clocks to be reset, and with the clock constraint φ defining the enabling condition for e.

4.2. Bounded liveness

Temporal properties are conventionally classified into safety and liveness properties. A liveness property is a property, stating that "something good will eventually happen", e.g., the program will terminate.
Nevertheless, this still insufficient to ensure correctness in case real-time deadlines must be observed. In reality, we need to establish that the property in question will hold within a certain upper time-bound. Thus, a bounded liveness property is a liveness property that comes with a maximal delay within which the "good thing" must happen. Several versions are available. We consider the more efficient one. Based on the method proposed in [15] in which time-bounded leads-to properties are reduced to simple safety properties, first the model under investigation is extended with a boolean variable b and an additional clock z. The boolean variable b must be initialized to false. Whenever φ starts to hold, b is set to true and the clock z is reset. When ψ commences to hold, b is set to false. Thus the truth-value of b indicates whether there is an obligation of ψ to hold in the future and z measures the accumulated time since this unfulfilled obligation started. The time-bounded leads-to property $\varphi \rightsquigarrow_{\leq t} \psi, t \subseteq \mathbb{R}$ is simply

obtained by verifying the safety property$\forall\Box(b \Rightarrow z \leq t)$.

Example: Intuitively, the formulae $\varphi \rightsquigarrow_{[2,3]} \psi$ express that for all the runs, always when φ holds, ψ holds in 2 to 3 units of time. InFigure 1, the formula $b \Rightarrow z \leq t$ holds for all states of one run. If this is true for all other runs, then the formulae $\varphi \rightsquigarrow_{[2,3]} \psi$ holds too.

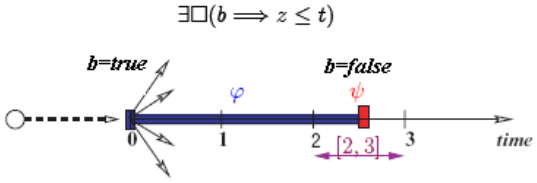


Figure 1Example of TCTL Formulae

4.3. Generic timed OWL-S verification approach

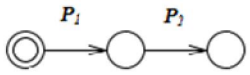
The paper describes how to model Web Services and determine whether the specification satisfies the bounded liveness property. In the following we present the different steps of the proposed approach:

- Step1: different web services are modeled in extended OWL-S language,
- Step2: automated composition is performed in respect of customer requirements,
- Step3: transformation of the composed service specification along with its imposed timing constraints into Timed Automata model,
- Step 4: specification of the formulae to verify and augment Timed Automata with a necessary variable as explained in section 4.2,
- Step 5: automatically verify the formula with Uppaal verifier,
- Step 6: If the property is not satisfied go to Step 1 to correct the OWL-S specification.

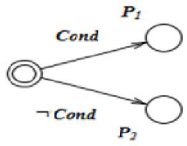
4.4. Timed OWL-S Timed Automata Transformation Rules

Timed OWL-S is taken to specify logical structures, more important, temporal aspects of service processes with rich semantic information. Transformed from the extended OWL-S process model, Timed Automata model is constructed to depict the structure and behavior of the (composite) service specified with service operation and time semantic information. With Uppaal tool, verification of temporal property can be conducted. In the following, we describe both untimed and timed transformation rules from Timed OWL-S to Timed Automata.

Sequence (P_1, P_2) Initial location is marked by a double circle.



If Cond then P_1 else P_2

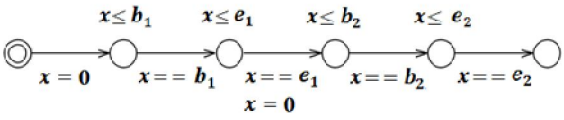


Split (P_1, P_2) and ***Split-join (P_1, P_2)*** Network (set) of Timed Automata allows expressing the parallelism. Edges labeled with complementary actions over the common channel Sync synchronize.

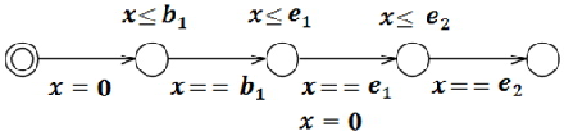
Repeat P While Cond

We can also obtain Repeat-Until by replacing $cond$ by $not\ cond$

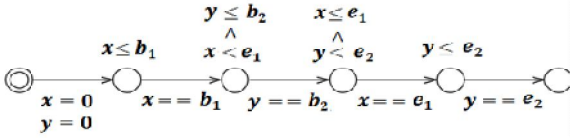
Before (d_1, d_2) We use the clock x to count time for$d_i, i \in \{1,2\}$. The guard $x == b_1$ (respectively $x == e_1$) marks the begin (respectively the end) of d_1. The invariant $x \leq b_1$ (respectively $x \leq e_1$) forces the system to leave once $x == b_1$ (respectively$x == e_1$). Analogously for d_2.



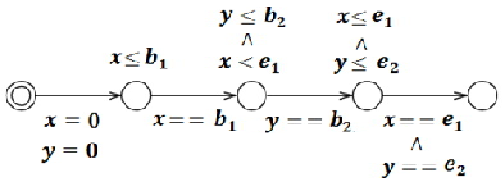
Meets (d_1, d_2)



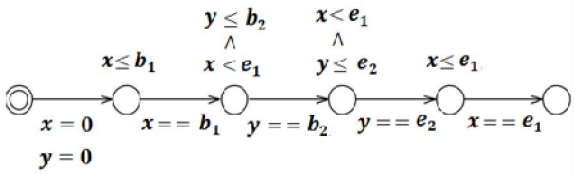
Overlaps (d_1, d_2) Since d_1 is not finished when d_2 starts, we use another clock y to count for d_2. Thus, x *and* y allow to express parallelism.



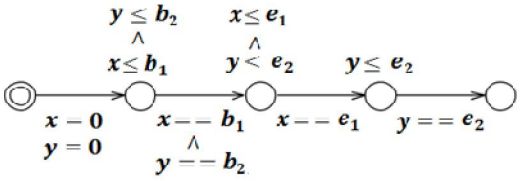
Finishes (d_1, d_2)



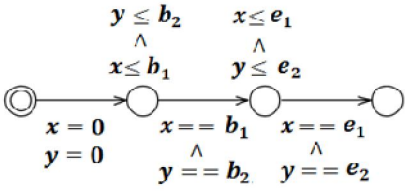
Covers (d_1, d_2)



Starts (d_1, d_2)



Equals (d_1, d_2)



5. Case Study: Multimodal Transport

Multimodal transport is being used across a wide range of government, it is generally considered as the most efficient way of handling an international door to door transport operation. This is so because multimodal transport allows combining in one voyage the specific advantages of each mode, such as the flexibility of road haulage, the larger capacity of railways and the lower costs of water transport in the best possible fashion. Multimodal transport also offers the shipper the possibility to rely on a single counterpart rather than having to deal with each and every modal specialist of the transport chain.

While multimodal transport seems to offer only benefits to all parties, shippers and service providers, it is also very difficult to achieve. It requires a thorough control over all the steps involved in international transport; this means extensive use of information technologies that can provide freedom to plan and operate to carriers and reliable liability regimes to customers.

In this case, we deal with the online shipment. Track-Ship is a fictitious service that offers online tracking and helps customers to take an appropriate decision to change transport strategy when it is necessary. For example, the transport mode chosen may have to change over time when delays happen. So let's consider this hypothetical Scenario. The supposed itinerary combines sea and railway transport. The departure date is in 5 days from registration and the arrival is in 10 days. On arrival, if there is no administrative problem, the railway transport is in 1day and take 2days to arrive at

destination. The delay due to administrative problem may take 2 days. In this case, customer has the choice to change to air transport or keep the previous plan, i.e.Sea-railway combined transport. A confirmation must be done no later than 1 to 2 days after shipment arrives in port, confirmation interval must be finished at the same instant as the interval corresponding to the administrative problem processing is finished. The airport departure is in 2 hours of administrative problem solving and the air transit time is not more than 1 hour. Finally, the goods are in stock within 1 day of arrival.

5.1. Modeling Temporal Behavior

The Track-Ship is a composite service. It contains three main services i.e. Sea, Railway and Air Services, each of them allows us to search information about available itineraries corresponding to our preferred needs. For example, the Air one is a service that makes it easy to find flights that meet our needs and planning travel. It includes a sequence of three operations i.e. GetDesiredDetails, SelectAvailableItinerary and Reservation. These three services are performed in parallel from which we use the control constraint "split" for the composition.

The OWL-S specification remains insufficient to describe adequately our scenario since the timed aspects are not taking account. For this purpose we add two output parameters: WaitTime, that specifies the time between reservation instant and departure instant, and ProcessTime, that specifies the time between departure instant and arrival instant.

It makes easy with the use of the entry sub-ontology of time; ProcessTime and WaitTime are defined as Intervals. To schedule the three itineraries we need to create ProcessTime and WaitTime subclasses corresponding to sea, railway and air information.

Also, two other classes of Interval type are created to describe intervals corresponding to administrative procedure (AdminProcedure) and duration allowed to change transport strategy (Change).

5.2. Service Composition

Now, it only remains for us the automated composition based on customer needs. Specifically, different timed schedules betweenthese intervals are met. For example, the temporal relation between ProcessTimeSea and AdminProcedure is constructed by the use of a restriction on the object property intMeets i.e. the expression "intMeets only AdminProcedure" to ProcessTimeSea class. We apply the same logic to the rest of the intervals.

```
<owl:Class rdf:about="http://www.isi.edu/-pan/
damltime/time-entry.owl#AdminProcedure">
 <rdfs:subClassOf rdf:resource="http://www.isi.edu/
 -pan /damltime/time-entry.owl# Interval"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="http://www.isi.ed
   -pan/damltime/time-entry.owl#intFinishes"/>
   <owl:allValuesFrom rdf:resource="http://www.isi
   edu/-pan/damltime/time- entry.owl#Change"/>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="http://www.isi.ed
   -pan/damltime/time-entry.owl#intMeets"/>
   <owl:allValuesFrom>
       <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="http://www
            time-entry.owl#WaitTimeAir"/>
            <rdf:Description rdf:about="http://www
            time-entry.owl#WaitTimeRailway"/>
```

Figure 3 AdminProcedure OWL Code

Figure 3shows the entire description of the class AdminProcedure.

Now, we need to ensure the correctness of the composed service i.e. goods are in stock within a fixed period of time for this

- In red, corresponds to the worst case where a customer chooses to keep the initial plan, i.e. sea-railway combined

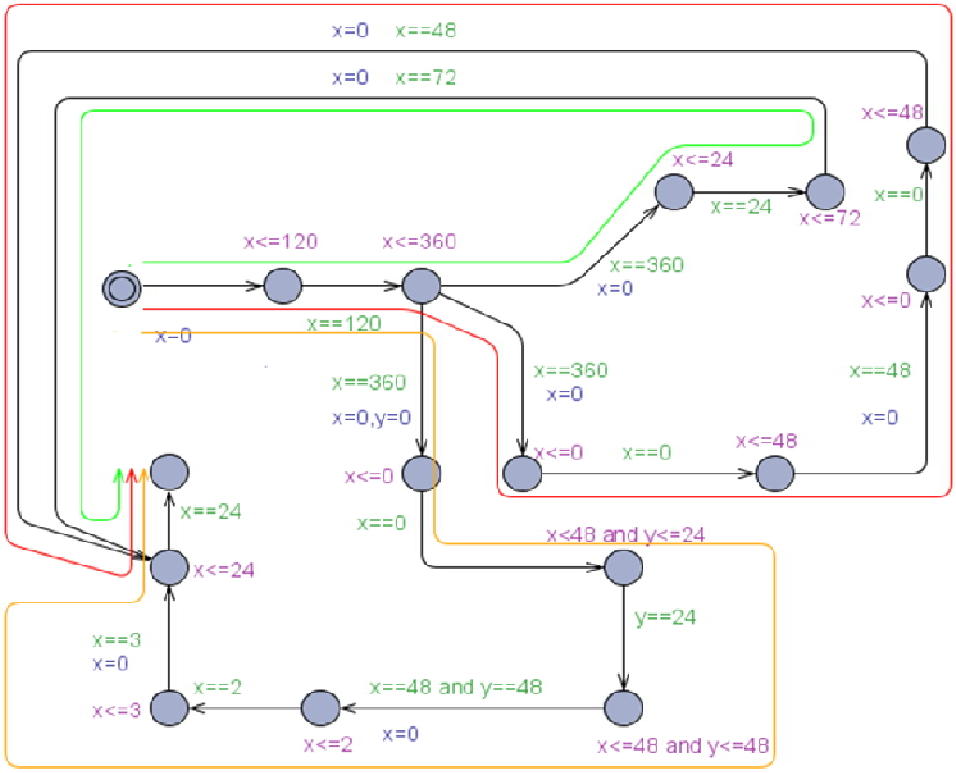


Figure 4 Generated Timed Automaton

case. In the following, we try to verify the property: "Goods are in stock within 19 days and 3 hours (i.e. 459 hours) from registration".

5.3. From timed OWL-S to timed automata

By applying transformation rules presented in Section 3.4, the Timed Automata is generated from the OWL-S specification. The generated Timed Automaton is presented in 4. We can easily distinguish three main runs:

- In green, the succession of actions corresponds to the sea-railway combined transport case;
- In orange, it corresponds to the situation where a customer chooses to change the transport strategy, to sea-air combined transport, in order to overtake lost time due to an administrative problem;

transport in spite of delays.

5.4. Formal Verification

Corresponding to [16], to specify the property to verify, the model under investigation must be extended with a boolean variable b and an additional clock Elapsed. The boolean variable b must be initialized to false. Whenever Registration is made, b is set to true and the clock Elapsed is reset. When the goods arrive, b is set to false. Thus the property "*Goods are in stock within 19 days and 3 hours (i.e. 459 hours) from registration*" is obtained by verifying the safety property ∀□(b⇒Elapsed≤459)

5.4.1. Uppaal verifier results

The verifier shows that the formula is not satisfied Hence, the selected travels cannot be accepted.

5.4.2. Uppaal simulator

The simulator allows us to views the counter example. In our case, it indicates the run corresponding to the situation when users choose to keep the first plan in spite of the delays.

Also, the simulator allows us to explore variable valuations (Figure 6). For example, when goods arrive by tracing the previous run, the clock Elapsed is in [456;480] and violate the condition Elapsed≤459. The lower bound of the interval corresponds to the arrival time at the airport and the upper bound corresponds to the time in which goods are available in stock. Moreover, the clock x counts time in which goods should be in stock after arriving. Contrariwise, the clock y is used to specify that the end of the confirmation interval (i.e. [24,48]) must correspond to the administrative problem resolution (i.e. [0,48]). Formally expressed as Finishes([0,48],[24,48]) used to the entry sub-ontology of time.

6. Conclusion And Future Work

This paper proposes a verification methodology for temporal properties of Semantic Web Service by Uppaal model checker. Taking account of time information, we use entry sub-ontology of time in conjunction with OWL-S for describing the temporal content of Web Services. Then we transform the OWL-S specification into Timed Automata, which is mandatory for an automated verification of Web Services. Transformation rules are projected and a case study is presented to show the applicability of our approach.

In future work, we aim to automate our proposed approach with a use of Model-Driven Engineering (MDE) tools for implementing transformation rules from Timed OWL-S ontology to Timed Automata model. In addition, we will extend the applicability of our approach to verify more properties. Finally, we also plan to verify more ambitious applications.

7. REFERENCES

[1] ter Beek, Maurice and Bucchiarone, Antonio and Gnesi, Stefania. A Survey on Service Composition Approaches:From Industrial Standards to Formal Methods. Technical Report 2006-TR-15. 2006

[2] Khadka, R., & Sapkota, B. An evaluation of dynamic web service composition approaches. [ed.] SciTePres. 2010.

[3] Diaz, G., Pardo, J. J., Cambronero, M. E., Valero, V., & Cuartero, F. Automatic translation of ws-cdl choreographies to timed automata . Formal Techniques for Computer Systems and Business Processes. 2005, pp. 230-242.

[4] Kazhamiakin, R., Pandya, P., et Pistore, M. Web Service Compositions. Engineering Service Compositions. 2005, p. 59.

[5] Chaochen, Z., Hoare, C. A. R., & Ravn, A. P. A calculus of durations. [ed.] Elsevier. 5, 1991, Information processing letters, Vol. 40, pp. 269-276.

[6] Dong, J. S., Liu, Y., Sun, J., & Zhang, X. Verification of computation orchestration via timed automata. Formal Methods and Software Engineering. 2006, pp. 226-245.

[7] Guermouche, N., & Godart, C. Timed conversational protocol based approach for web services analysis. Service-Oriented Computing. 2010, pp. 603-611.

[8] Liu, R., Hu, C., Zhao, C., & Gao, Z. Verification for time consistency of Web service flow. Computer and Information Science. May 2008, pp. 624-629.

[9] Emilia Cambronero, M., Díaz, G., Valero, V., & Martínez, E. Validation and verification of Web services choreographies by using timed automata. 1, 2011, Journal of Logic and Algebraic Programming, Vol. 80, pp. 25-49.

[10] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., ... & Sycara, K. Bringing semantics to web services: The OWL-S approach. [ed.] Springer. Semantic Web Services and Web Process Composition. 2005, pp. 26-42.

[11] Hobbs, J. R., & Pan, F.An ontology of time for the semantic web. [ed.] ACM. 1, 2004, ACM Transactions on Asian Language Information Processing (TALIP), Vol. 3, pp. 66-85.

[12] Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., & Schnoebelen, P. Systems and software verification: model-checking techniques and tools. 2010.

[13] Alur, R., Courcoubetis, C., & Dill, D. Model-checking in dense real-time. 1993. pp. 2-34.

[14] Alur, R., Dill, D. L. A theory of timed automata. Theoretical computer science . 1994. pp. 183-235.

[15] Lindahl, M., Pettersson, P., & Yi, W. Formal design and analysis of a gear controller. [ed.] Springer. Tools and Algorithms for the Construction and Analysis of Systems. 1998, pp. 281-297.

[16] Boumaza, A., Maamri, R. Bounded liveness in semantic web services. ACIT. December 2012, 13, pp. 602-610.