Passive Testing of Timed Distributed Systems *

César Andrés[1], M. Emilia Cambronero[2], and Manuel Núñez[1]
c.andres@fdi.ucm.es,emicp@dsi.uclm.es, mn@sip.ucm.es

Dept. Sistemas Informáticos y Computación Universidad Complutense de Madrid, 28040 Madrid, Spain

Abstract. This paper presents a formal framework to perform passive testing of service-oriented systems. Our approach uses the historical interaction files between web services to check the absence of faults. It uses a set of properties, that we call *invariants*, to represent the most relevant expected behavior of the web services under test. Intuitively, an invariant expresses the fact that each time the system under test performs a given sequence of actions, it must exhibit a behavior reflected in the invariant. Invariants can be defined from a local point of view, that is, to check properties of isolated web services, and from a global point of view, that is, to check web service interaction properties. In order to increase applicability and adaption to a real environment, we assume that we do not have a global log. We show how to use local logs (recorded in each web service) in order to check local properties and how to combine them in order to check global properties.

Keywords: Passive Testing, Service Oriented Systems, Monitoring.

1 Introduction

The complexity of current systems, the large number of people working on them, and the number of different modules that interact with each other, make it difficult to assess their correctness. *Testing techniques* allow us to provide a degree of confidence in the correctness of a system. Testing techniques can be combined with the use of formal methods in order to semi-automatically perform some of the tasks involved in testing. The application of formal testing techniques to check the correctness of a system requires to identify its *critical* aspects, that is, those characteristics that will make the difference between correct and incorrect behaviors. While the relevant aspects of some systems only concern *what* they do, in some other systems it is equally relevant *how* they do what they do. Thus, during the last years formal testing techniques also deal with non-functional properties. In this work we will focus on web services, which relate technologies for supporting *Service-Oriented Computing*, and consider systems that contain temporal restrictions, being already several proposals for timed testing (see for example, [**?**,**?**], for some proposals to test timed systems).

* Research partially supported by the Spanish MEC project TESIS TIN2009-14312-C02.

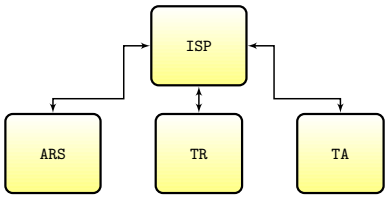

Fig. 1. Structure of the web services monitoring.

In testing there are usually two approaches: *Active* and *passive testing*. The main difference between them is how a tester can interact with the Implementation Under Test (IUT). In the active paradigm the tester is allowed to apply any set of tests to the IUT. In passive testing, the tester is only an observer of the IUT, and he has to provide a degree of confidence of the system, only taking into account the monitored traces.

In this paper we present a formal passive testing methodology to test web services with temporal restrictions. In our passive testing paradigm, testers provide a set of *invariants* that represents the most relevant properties that they would like to check against the logs. Our approach makes use of the ideas presented in [**?**], a timed extension of the framework defined in [**?**], to define invariants, in [**?**] to define orchestrator and choreography behaviors, and in [**?**], our previous approach to test web services using invariants without considering a timed environment. In this paper, a Service-Oriented Scenario consists of a large number of services that interact with each other. For example, let ARS, TR and TA be three web services that are interacting. When this set of services is monitored according to [**?**], we have the scheme presented in Figure **??**. The set of requests of the web services is conveyed across a network operated by an ISP. When the monitoring task is performed, a set of four logs is provided, corresponding to the ARS, ISP, TR, and TA web services, respectively. Passive testing techniques make use of invariants to represent the properties that we like to check against these logs. According to the previous monitoring result, we are able to observe all traces of the system composition if we monitor the ISP system. In this paper, however we do not assume that we can monitor the ISP system, because this task is difficult and very costly, due to the fact that the ISP can be composed of a set of ISPs and its internal structure becomes very complex. We also assume that we are not provided with a *global clock* to mark the time stamps associated to actions. Furthermore, the clocks of the set of services can differ from each other. Within this scenario, we show how we can define *local invariants* that involve the local logs of the services, and how we can solve the absence of the global log, by combining the set of local logs, that can be used to check *global invariants*.

Our invariants can be seen as the SLAs presented in [**?**], but there are some relevant differences. In [**?**] the set of SLAs are formulas that are extracted from the most frequent patterns of the specification. So, these formulas do not contradict the specification. In our approach, we assume that invariants are provided

either by a tester or by using data mining techniques [**?**]. So, the correctness of the invariants must be checked with respect to the specification. Most importantly we do not need to exchange additional information between web services since we have a decentralized approach.

There already exist several approaches to study the integration of formal testing in web services, providing formal machinery for representing and analyzing the behavior of communicating concurrent/distributed systems. Next we briefly review some previous work related to our framework. In [**?**] an automatic back-box testing approach for WS-BPEL orchestrations was presented, which was based on translation into Symbolic Transition System and Symbolic Execution. This approach supports the rich XML-based data types used in web services without suffering from state explosion problems. This work was inspired in a previous work on symbolic testing [**?**], where the authors showed a modeling and testing approach focused on detecting failures, supporting conformance, and reducing drastically the effort to design test cases, validate test coverage, and execute test cases. In [**?**] a methodology to automatically generate test cases was presented. The authors combine coverage of web services operations with data-driven test case generation. These test cases were derived from WSDL [**?**] descriptions. For that purpose, they used two tools: soapUI [**?**] and TAXI [**?**]. The first one generates stubs of SOAP calls for the operations declared in a WSDL file. The other one facilitates the automated generation of XML instances from an XML Schema. In [**?**] the authors present different mechanisms to collect traces. They also study the differences between online and offline monitoring, being the main difference that the testing algorithms in online monitoring are adapted to analyze the information as soon as possible, so a huge amount of computational resources is needed. Their methodology differs from our approach since they use the specification to check the correctness of the traces, while we might have only invariants, and they record *global* traces while we can operate at a *local* level. The work reported in [**?**] presents how Service Level Agreements (`SLAs`) among a web service provider, a web service user, and an Internet Service Provider (`ISP`) should be arranged in a manner that they can be monitored. The work presented in [**?**] deals with the problem of auto extracting a set of services that conform to a given choreography.

The rest of this paper is structured as follows. First, Section **??** presents our formal framework to represent web services choreographies, and orchestrations. In Section **??** we present how to define local and global invariants. Finally, in Section **??** we present our conclusions and some lines for future work.

2 Preliminaries

In this section we present our formalism to define web services and choreography models. We follow the ideas underlying the definition of orchestration and choreography model behaviors presented in [**?**]. However, instead of Finite State Machines, we use *Timed Automata*, with a finite set of clocks over a dense time domain, to represent orchestrations. Since we will not use most of the technical

machinery behind Timed Automata, the reader is referred to [?]. The internal behavior of a *web service* is given by a Timed Automaton where the clock domain is defined in $\mathbb{R}_+$. The choice of a next state in the automaton does not only depend on the action, but also on the timed constraints associated to each transition. Only when the time condition is satisfied by the current values of the clocks, the transition can be triggered. We assume that the communication between systems is asymmetric.

Definition 1. A *clock* is a variable c in $\mathbb{R}_+$. A set of clocks will be denoted by $\mathcal{C}$. A *timed constraint* φ on $\mathcal{C}$ is defined by the following EBNF:

$$\varphi ::= \varphi \wedge \varphi \mid c \leq t \mid c < t \mid \neg\varphi$$

where $c \in \mathcal{C}$ and $t \in \mathbb{R}_+$. The set of all timed constraints over a set $\mathcal{C}$ of clocks is denoted by $\phi(\mathcal{C})$.

A *clock valuation* ν for a set $\mathcal{C}$ of clocks assigns a real value to each of them. For $t \in \mathbb{R}_+$, the expression $\nu + t$ denotes the clock valuation which maps every clock $c \in \mathcal{C}$ to the value $\nu(c) + t$. For a set of clocks $\mathcal{Y} \subseteq \mathcal{C}$, the expression $\nu[\mathcal{Y} := 0]$ denotes the clock valuation for $\mathcal{C}$ which assigns 0 to each $c \in \mathcal{Y}$ and agrees with ν over the rest of the clocks. The set of all clock valuations is denoted by $\Omega(\mathcal{C})$.

Let ν be a clock valuation and φ be a timed constraint. We write $\varphi \vdash \nu$ iff ν holds φ; $\varphi \nvdash \nu$ denotes that ν does not hold φ. □

Next we define a *web service*. The internal behavior of a web service, in terms of its interaction with other web services, is represented by a *Timed Automaton*.

Definition 2. We call any value $t \in \mathbb{R}_+$ a *fixed time value*. For all $t \in \mathbb{R}_+$ we have both $t < \infty$ and $t + \infty = \infty$. We say that $\hat{p} = [p_1, p_2]$ is a *time interval* if $p_1 \in \mathbb{R}_+$, $p_2 \in \mathbb{R}_+ \cup \{\infty\}$, and $p_1 \leq p_2$. We consider that $\mathcal{I}_{\mathbb{R}_+}$ denotes the set of time intervals.

Along this paper ID denotes the set of web service identifiers. A *web service* is a tuple $\mathcal{A} = (id, \mathcal{S}, s_0, \Sigma, \mathcal{C}, \mathcal{Z}, \mathcal{E})$ where $id \in ID$ is the identifier of the service, $\mathcal{S}$ is a finite set of states, $s_0 \in \mathcal{S}$ is the initial state, Σ is the alphabet of actions, $\mathcal{C}$ is a finite set of clocks, $\mathcal{Z} : \mathcal{S} \to \phi(\mathcal{C})$ associates a time condition to each state, and $\mathcal{E} \subseteq \mathcal{S} \times \{id\} \times \Sigma \times ID \times \phi(\mathcal{C}) \times \wp(\mathcal{C}) \times \mathcal{S}$ is the set of transitions. We will consider that Σ is partitioned into two (disjoint) sets of *inputs* denoted by $\mathcal{I}$, preceded by `?`, and *outputs* denoted by $\mathcal{O}$, preceded by `!`. Along this paper Σ^{ID} denotes the set $ID \times \Sigma \times ID$.

We overload the $\vdash$ symbol. Let $s \in \mathcal{S}$ and $\nu \in \Omega(\mathcal{C})$. We denote by $s \vdash \nu$ the fact that ν holds $\mathcal{Z}(s)$ (resp. $s \nvdash \nu$ represents that ν does not hold $\mathcal{Z}(s)$). Let $e = (s, id, \alpha, id', \varphi, \mathcal{Y}, s') \in \mathcal{E}$. We denote by $e \vdash \nu$ the fact that ν holds φ (resp. $e \nvdash \nu$ represents that ν does not hold φ). □

Intuitively, a transition $(s, id, \alpha, id', \varphi, \mathcal{Y}, s')$ indicates that if the system is at state s and the current valuation of the clocks holds φ, then the system moves to the state s' performing the action α from id to id' and resetting the clocks

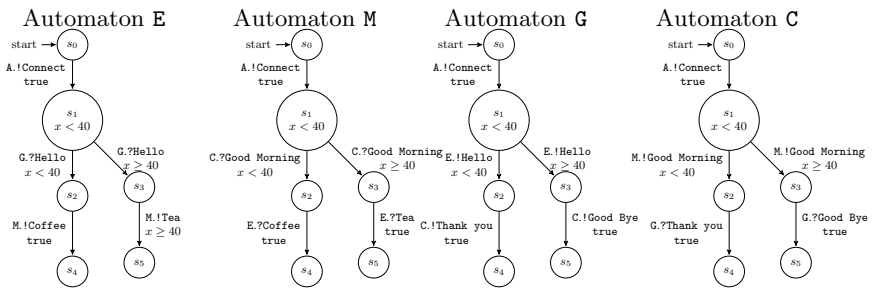

Fig. 2. Example of four web services.

in $\mathcal{Y}$. In other words if we consider $e_1 = (s, id, ?\alpha, id_b, \varphi, \mathcal{Y}, s')$ then the action α is emitted from id to id_b, and if we consider $e_2 = (s, id, !\alpha, id_b, \varphi, \mathcal{Y}, s')$ then the action α is received on id from id_b. For each state s, $\mathcal{Z}(s)$ represents a timed constraint for s, that is, the system can remain in s while the current valuation of the clocks holds $\mathcal{Z}(s)$. We will assume the following usual condition on timed automata: For all $s \in \mathcal{S}$ and all valuation $\nu \in \Omega(\mathcal{C})$ if $s \nvdash \nu$ then there exists at least a transition $e = (s, id, \alpha, id', \varphi, \mathcal{Y}, s') \in \mathcal{E}$ with $e \vdash \nu$. This property allows to leave a state once the restrictions on clocks do not hold in that state.

Example 1. Next we present a small running example to explain the previous concepts. Let us consider the set of four web services represented in Figure **??**. In this example we will consider that we have only one clock for each web service, and it is set to 0 in the transition that reach the state s_1. The time constrains represented in some states, for example $x < 40$ in s_1 of `G`, represent the time that this automaton is allowed to be in the state. In the transitions are represented two items. The first one is a pair composed of the id of the web service that interacts with this automaton and the action that they exchange. The second one represents the condition to trigger this transition.

These automata are communicating each others. All of them start receiving `Connect` from another web service, called `A`, in order to be synchronized. After this, each one has its own behaviour. Let us remark that the time between receiving `Connect`, and to perform next action will decide the future behaviour of the web service. For example if we consider that this time is less than 40 time unit and `M` sends `Good Morning` to `C`. Then after any elapsed of time it only can send to `E` the message `Tea`. □

The *semantics* of a web service is given by translating it into a *labeled transition system* with an uncountably number of states. Let us remark that, in general, we will not construct the associated labeled transition system; we will use it to reason about the traces of the corresponding timed automaton.

Definition 3. A *Labeled Transition System*, in short `LTS`, is defined by a tuple $\mathcal{M} = (ID, \mathcal{Q}, q_0, \Sigma, \rightarrow)$, where ID is a set of web service identifiers, $\mathcal{Q}$ is a set of states, $q_0 \in \mathcal{Q}$ is the initial state, Σ is the alphabet of actions, and the relation $\rightarrow \subseteq \mathcal{Q} \times \Sigma^{ID} \cup \mathbb{R}_+ \times \mathcal{Q}$ represents the set of transitions.

Let $\mathcal{A} = (id, \mathcal{S}, s_0, \Sigma, \mathcal{C}, \mathcal{Z}, \mathcal{E})$ be a web service. Its *semantics* is defined by its associated LTS, $\mathcal{A}_{\mathcal{M}} = (ID, \mathcal{Q}, q_0, \Sigma, \rightarrow)$, where $\mathcal{Q} = \{(s, \nu) \mid s \in \mathcal{S} \wedge \nu \in \Omega(\mathcal{C}) \wedge s \vdash \nu\}$, $q_0 = (s_0, \nu_0)$, being $\nu_0(c) = 0$ for all $c \in \mathcal{C}$, and we apply two rules in order to generate the elements of $\rightarrow$. For all $(s, \nu) \in \mathcal{Q}$ we have:

- If for all $0 \leq t' \leq t$ we have $s \vdash (\nu + t')$, then $((s, \nu), t, (s, \nu + t)) \in \rightarrow$.
- If $e \vdash \nu$, for $e = (s, id, \alpha, id', \varphi, \mathcal{Y}, s') \in \mathcal{E}$, then $((s, \nu), (id, \alpha, id'), (s', \nu[\mathcal{Y} := 0])) \in \rightarrow$.

In addition, we consider the following conditions: (a) If we have $q \xrightarrow{t} q'$ and $q' \xrightarrow{t'} q''$, then we also have $q \xrightarrow{t+t'} q''$ and (b) if $q \xrightarrow{0} q'$ then $q = q'$, that is, a passage of 0 time units does not change the state. The set of all LTS associated with web services will be denoted by SetLTS. □

Next, we introduce the notion of *visible trace*, or simply *trace*. As usual, a trace is a sequence of visible actions and time values.

Definition 4. Let $\mathcal{M} = (ID, \mathcal{Q}, q_0, \Sigma, \rightarrow)$ be a LTS, $\vartheta_1, \ldots, \vartheta_n \in \Sigma^{ID}$, and $t_1, \ldots, t_{n-1} \in \mathbb{R}_+$. We say that $\sigma = \langle \vartheta_1, t_1, \vartheta_2, t_2, \ldots, t_{n-1}, \vartheta_n \rangle$, with $n > 0$, is a visible trace, or simply trace, of $\mathcal{M}$ if there exits the transitions (q_1, ϑ_1, q_2), $(q_2, t_1, q_3), \ldots, (q_{2*n-2}, t_{n-1}, q_{2*n-1}), (q_{2*n-1}, \vartheta_n, q_{2*n}) \in \rightarrow$. We will denote by $\mathtt{NT}(\mathcal{M})$ the set of all visible traces.

We define the function TT as the sum of all time values of a normalized visible trace, that is $\mathtt{TT}(\sigma) = \sum_{i=1}^{n-1} t_i$. We denote by $\sigma_{<<} \subseteq$ SetNVT the set of all subsequences of σ that are visible traces. □

We will usually consider normalized visible traces since this is what we observe from the execution of a system. We cannot observe either internal activity (that is, the performance of internal actions) or different passages of time associated to different transitions. Logs recorded from a IUT, will look like visible traces.

Example 2. Let us consider the web services presented in Figure **??**, and M.LOG = ⟨A.!Connect, 60, C.?Good Morning, 50, E.?Tea, ⟩ be a local log recorded in the web service M. Intuitively, this log represents that A sends Connect to M in order to synchronize with the others web services. Then, after 60 time units, M has sent to C the message Good Morning, and after 50 local time units, it has sent to E the action Tea. □

Next we introduce our formalism to represent *choreographies*. Contrarily to systems of orchestrations, choreographies focus on representing the interaction of web services as a whole. Thus a single machine, instead of the composition of several machines, is considered. The choreography model also is a timed automata, but there are the following differences with respect to web services model: The first one is that there exists only one clock, that is called global clock; the second one is that in the transitions is represented the interaction of the web services, and the third one is that the valuation of the global clock in the initial state is 0, and it can not be reseted.

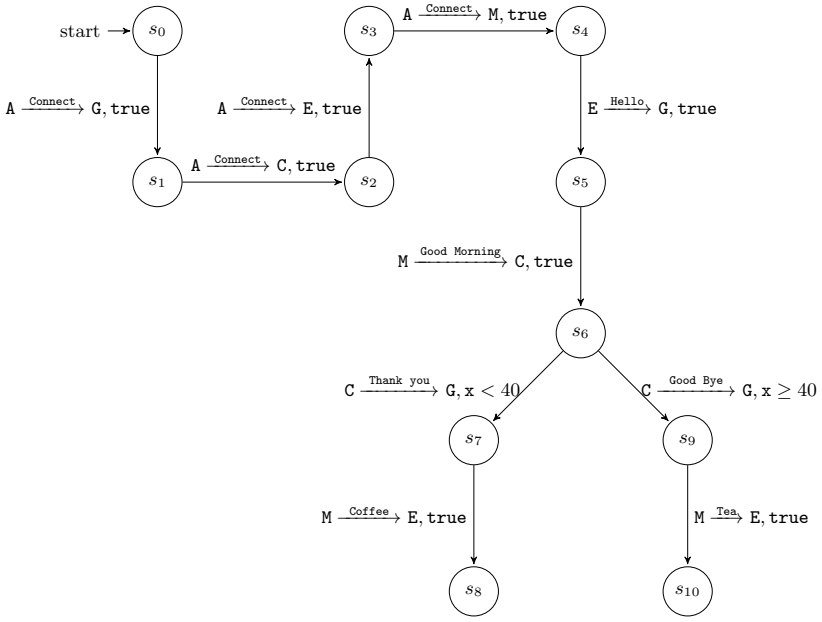

Fig. 3. Choreography of C, E, G and M.

Definition 5. A *choreography machine* is a tuple $\mathcal{D} = (\mathcal{S}, \Sigma, ID, s_0, \{x\}, \mathcal{Z}, \mathcal{T})$ where $\mathcal{S}$ denotes the set of states, Σ is the set of messages, ID is the set of web service identifiers, $s_0 \in \mathcal{S}$ is the initial state, $\{x\}$ is a clock called global clock, $\mathcal{Z} : \mathcal{S} \to \phi(\{x\})$ associates a time condition to each state, and $\mathcal{T} \subseteq \mathcal{S} \times \Sigma^{ID} \times \mathcal{S}$ is the set of transitions. The initial valuation of $\{x\}$ is 0. And this clock can never be reseted. □

The notions of traces, and the transformation of the choreography into its LTS associated are similar that the one presented for the web services. Concerning choreography machines, transitions are tuples $(s, id, \alpha, id', \varphi, s')$ where $s, s' \in \mathcal{S}$ are the initial and final states, α is the message, and $id, id' \in ID$ are the sender and the addressee of the message, respectively. Next, let us introduce the idea of choreography with the following example.

Example 3. Let us consider the choreography represented in Figure **??**. In this model is denoted the global behaviour of the web services presented in Figure **??**. Let us denote that the initial valuation of the global clock, by means x is 0.

In the figure, the transition $s_4\mathtt{E} \xrightarrow{\mathtt{Hello}} \mathtt{G}, \mathtt{true}, s_5$ represents that the web service E will send the message Hello to G. The value true means that there not exists any time constrain associated with this transition. This choreography is a reduced graph of the complete choreography of these web services. In the complete one we will have to increase with the permutation of all possible interaction of the web services from s_0 to s_6, from s_6 to s_8 and from s_6 to s_{10}. □

3 Invariants

In this section we introduce the notion of *invariant*. Invariants are used in our approach to represent the properties that we would like to check against the logs extracted from the `IUT`. The notion of invariant being *correct* with respect to a specification means that if the invariant detects a mismatch, then the implementation that has generated this log is incorrect with respect to the specification. First, after producing a set of invariants and before checking them against the log, they must be checked against the specification; otherwise, we might have an invariant which indicates an erroneous behavior that does not violate the requirements expressed in the specification. Another possibility would be to consider that invariants are correct *by definition*. In this case a mismatch will automatically imply that a fault was detected.

We present two different kinds of invariants: *Local invariants* and *global invariants*. The first type is used to express properties of isolated web services, while the second one, will use the combination of all isolated logs, to check some significant properties at the system level.

3.1 Local Invariants

Definition 6. Let $\mathcal{A} = (id_a, \mathcal{S}, s_0, \Sigma, \mathcal{C}, \mathcal{Z}, \mathcal{E})$ be a web service. We say that the sequence ϕ_{id_a} is a *local invariant* for $\mathcal{A}$ if it is defined according to the following EBNF:

$$\begin{aligned} \phi_{id_a} &::= \texttt{Body} \mapsto \texttt{Consequent} \\ \texttt{Body} &::= \vartheta/\hat{p}, \texttt{Body} \mid \star/\hat{p}, \texttt{Body} \mid \vartheta'/\hat{p} \\ \texttt{Consequent} &::= O \rhd \hat{p} \end{aligned}$$

In this expression we consider that $\hat{p} \in \mathcal{I}_{\mathbb{R}_+}$, $\vartheta' \in \{id_a\} \times \Sigma \times ID$, $\vartheta \in \{id_a\} \times \Sigma \cup \{?\} \times ID$, and $O \subseteq \{id_a\} \times \Sigma \times ID$.

Let $\phi_{id} = \mathcal{P} \mapsto \mathcal{R}$ be a local invariant. We define the functions $\texttt{Body}(\phi_{id}) = \mathcal{P}$ and $\texttt{Consequent}(\phi_{id}) = \mathcal{R}$. The set of all invariants for a set of web services identifiers ID is denoted by Φ_{ID}, and the set of all bodies of these invariants is denoted by Φ_{ID}^{body}. □

Let us remark that time conditions established in invariants are given by intervals. However, web services in our formalism present fix time. Intervals represent the idea that it can be admissible that the execution of a task sometimes takes more time than expected: If most of the times the task is performed on time, a small number of delays can be tolerated. Moreover, another reason for the tester to allow imprecisions is that the artifacts measuring time while testing a system might not be as precise as desirable. In this case, an apparent wrong behavior due to bad timing can be in fact correct since it may happen that the *clocks* are not working properly.

In our framework, the symbol ? can replace any action while the symbol $\star$ can replace a sequence of actions not containing the first action symbol that appears in the part of the invariant that follows it. Intuitively, the EBNF expresses that a local invariant is either a sequence of symbols where each component, but

the last one, is either or an expression $id_a, \alpha, id_b/\hat{p}$, with id_a and id_b being web services identifier, with α being an action or the wildcard character ?, and $\hat{p}$ being a timed interval, or an expression $\star/\hat{p}$.

There are two restrictions to this rule: a local invariant cannot contain two consecutive components $\star/[p_1, p_2]$ and $\star/[q_1, q_2]$ since this situation could be simulated by means of the expression $\star/[p_1 + q_1, p_2 + q_2]$, and a local invariant cannot present a component of the form $\star/\hat{p}$ followed by a wildcard character ?, that is, the action of the next component must belong to Σ. The last component, corresponding to the expression $\vartheta'/\hat{p} \mapsto O \rhd \hat{q}$, is composed of two web service identifiers associated with an action, that is ϑ', followed by a timed interval, and followed by a set of triples identifier/actions/identifier and another time interval.

When we check a log with respect to a local invariant, first we check if the log *matches* the body of the invariant. When we find a sequence that matches, then we check the correctness of this sequence. The correctness of a sequence can have the usual three valued valuations: *correct*, *incorrect* and *inconclusive*. The result is returned inconclusive if the trace never matches the body of the invariant. An invariant can detect an error with respect to two restrictions. These are represented in the consequent part of the invariant, that is $O \rhd \hat{q}$. The first requirement is given by O, and it is associated to the last term of the log. It means that if a log $\langle \vartheta_1, \ldots, \vartheta_n \rangle$ matches the body of the invariant, then last component, by means ϑ_n, must belong to O. Meanwhile $\hat{q}$ means that the sum of all time values presented in the log must belong to this interval, that is $\mathtt{TT}(\langle \vartheta_1, \ldots, \vartheta_n \rangle) \in \hat{q}$.

Definition 7. Let $\vartheta = \langle id_a, \alpha, id_b \rangle$, $\vartheta' = \langle id'_a, \alpha', id'_b \rangle \in \Sigma^{ID}$, be two items of a normalized visible trace, $t \in \mathbb{R}_+$ be a time value and $\hat{p} \in \mathcal{I}_{\mathbb{R}_+}$ be an interval. We define the function compare, denoted by $\mathtt{c} : (\Sigma^{ID} \times \mathbb{R}_+) \times (\Sigma^{ID} \times \mathcal{I}_{\mathbb{R}_+}) \mapsto \mathit{Bool}$ as follow:

$$\mathtt{c}(\langle \vartheta, t \rangle, (\vartheta'/\hat{p})) = ((id_a = id'_a) \wedge (id_b = id'_b) \wedge (\alpha = \alpha') \wedge (t \in \hat{p}))$$

Let $\sigma = \langle \vartheta_1, t_1, \ldots, \vartheta_n, t_n \rangle$, be a normalized visible trace, with $n > 0$, and $\mu = (\vartheta'/\hat{p}, \ldots, \vartheta'_m/\hat{p}_m)$, with $m > 0$, be a body of an invariant. Let $\mathtt{Match} : \mathtt{SetNVT} \times \Phi_{ID}^{body} \mapsto \mathit{Bool}$ be a function that computes if a normalized visible trace and an invariant *matches*. Formally we define $\mathtt{Match}(\sigma, \mu)$ as:

$$\begin{cases} \mathtt{false} & \text{if } n > 1 \wedge m = 1 \vee n = 1 \wedge m > 1 \\ \mathtt{c}(\langle \vartheta_1, t_1 \rangle, (\vartheta'_1/\hat{p}_1)) & \text{if } \sigma = \langle \vartheta_1, t_1, \vartheta_2 \rangle \wedge \mu = (\vartheta'_1/\hat{p}_1) \\ \mathtt{Match}(\langle \vartheta_2, \ldots, \vartheta_n \rangle, (\vartheta'_2/\hat{p}_2, \ldots, \vartheta'_m/\hat{p}_m)) & \text{if } n > 1 \wedge m > 1 \wedge \mathtt{c}(\langle \vartheta_1, t_1 \rangle, (\vartheta'_1/\hat{p}_1)) \\ \mathtt{false} & \text{if } n > 1 \wedge m > 1 \wedge \neg\mathtt{c}(\langle \vartheta_1, t_1 \rangle, (\vartheta'_1/\hat{p}_1)) \\ \mathtt{M'}(\sigma, \mu', \hat{q}, 0) & \text{if } n > 2 \wedge \mu = (\star/\hat{q}, \ldots, \vartheta'_m, \hat{p}_m) \end{cases}$$

where $\mathtt{M'} : \mathtt{SetNVT} \times \Phi_{ID}^{body} \times \mathcal{I}_{\mathbb{R}_+} \times \mathbb{R}_+ \to \mathit{Bool}$ is an auxiliary function used to compute the appearance of the wildcard $\star$.

Let $\mu = (\vartheta'_1/\hat{p}'_1, \ldots \vartheta'_m/\hat{p}'_m)$, with $m > 0$ and $\vartheta'_1 = (id'_a, \alpha', id')$, be the body of an invariant, $\sigma = \langle \vartheta_1, t_1, \ldots, \vartheta_n \rangle$, with $n > 0$ and $\vartheta_1 = (id_a, \alpha, id)$, be a normalized visible trace, $\hat{q} = [q_1, q_2] \in \mathcal{I}_{\mathbb{R}_+}$ be timed interval, and $t \in \mathbb{R}_+$. Formally, we define $\texttt{M'}(\sigma, \mu, \hat{q}, t)$ as

$$\begin{cases} \texttt{false} & \text{if } t > q_2 \vee n = 1 \vee (id_a = id'_a \wedge t \notin \hat{q}) \\ \texttt{Match}(\sigma, \mu) & \text{if } id_a = id'_a \wedge t \in \hat{q} \\ \texttt{M'}(\langle \vartheta_2, t_2, \ldots, \vartheta_n \rangle, \mu, [q_1, q_2], t + t_1) & \text{if } t \leq q_2 \wedge id_a \neq id'_a \end{cases}$$

Let ϕ be a local invariant and $\sigma = \langle \vartheta_1, t_1, \ldots, \vartheta_n \rangle$ be a trace. We say that σ is *inconclusive* with respect to ϕ if $\forall \sigma' \in \sigma_{<<}$ we have that $\texttt{Match}(\sigma', \texttt{Body}(\phi))$ does not hold. Let $O \rhd \hat{q} = \texttt{Consequent}(\phi)$, we say that σ is *correct* with respect to ϕ if $\forall \sigma' = \langle \vartheta_b, t_b, \ldots, \vartheta_r \rangle \in \sigma_{<<}$, with $1 \leq b < r \leq n$, if $\texttt{Match}(\sigma', \texttt{Body}(\phi))$ then we have that $\vartheta_r \in O$ and $\texttt{TT}(\langle \vartheta_1, t_1, \ldots, \vartheta_r \rangle) \in \hat{q}$.

We say that σ is *not correct* with respect to ϕ if $\exists \sigma' = \langle \vartheta_b, t_b, \ldots, \vartheta_r \rangle \in \sigma_{<<}$, with $1 \leq b < r \leq n$, if $\texttt{Match}(\sigma', \texttt{Body}(\phi))$ then we have that $\vartheta_r \notin O$ or $\texttt{TT}(\langle \vartheta_1, t_1, \ldots, \vartheta_r \rangle) \notin \hat{q}$.

We denote by $\sigma \diamond \phi$ the fact that σ is correct with respect to ϕ, alternatively $\sigma \neg \diamond \phi$ denotes that is erroneous. □

Next, we will illustrate the semantics of a local invariant by using an example. Let $\phi = id, \alpha, id'/\hat{p},\ \star/\hat{p}_\star,\ id, \alpha', id''/\hat{p}' \mapsto O \rhd \hat{q}$ be a local invariant. This property, with respect to a recorded trace, means that **if** we observe the action α from id to id' in a time belonging to the interval $\hat{p}$, followed by a (possibly empty) sequence of actions without occurrence of the action α', then **if** we observe the input symbol α' from id to id'', and the lapse of time between the performance of the action α and input α' belongs to the interval $\hat{p_\star}$ **then** the head of the invariant must hold. This means that α' must be followed by a triple id-action-id belonging to the set O with an associated time value belonging to $\hat{p}'$. The interval $\hat{q}$ makes reference to the total time that the system must spend to perform the whole trace. Let us remark that an invariant can only detect an error if the body of the invariant, that is the part of the invariant previous to $\mapsto$ symbol, matches the log and, either the functional restriction does not match or any temporal requirement does not match. When an invariant is provided by a tester, before using it to check the correctness of a log, we may ensure that this invariant does not contradict what is represented in the specification model, that is, the invariant has to be *correct* with respect to the specification.

Definition 8. We say that an invariant ϕ is *correct* with respect to a web service $\mathcal{A}$, being $\mathcal{A}_\mathcal{M}$ the LTS associated with $\mathcal{A}$, if the following two conditions hold: For all $\sigma \in \texttt{NT}(\mathcal{A}_\mathcal{M})$ we have both conditions: or $\sigma \diamond \phi$ or σ is inconclusive with respect to ϕ, and there exits $\sigma \in \texttt{NT}(\mathcal{A}_\mathcal{M})$ with $\sigma \diamond \phi$. □

Example 4. Let us consider the web services specification presented in Figure **??**. Next we show how we can express some properties with timed invariants for the

$$\phi_{\mathtt{E}1} = \begin{matrix}\mathtt{E}, \mathtt{!Connect}, \mathtt{A}[0,39], \\ \mathtt{E}, \mathtt{?Hello}, \mathtt{G}[0,\infty]\end{matrix} \mapsto \{(\mathtt{E}, \mathtt{!Coffee}, \mathtt{M})\} \rhd [0,\infty]$$

$$\phi_{\mathtt{E}2} = \mathtt{E}, \mathtt{!Connect}, \mathtt{A}[0,39], \mapsto \{(\mathtt{E}, \mathtt{?Hello}, \mathtt{G})\} \rhd [0,39]$$

$$\phi_{\mathtt{E}3} = \begin{matrix}\mathtt{E}, \mathtt{!Connect}, \mathtt{A}[41,\infty], \\ \mathtt{E}, \mathtt{?Hello}, \mathtt{G}[0,\infty]\end{matrix} \mapsto \{(\mathtt{E}, \mathtt{!Tea}, \mathtt{M})\} \rhd [41,\infty]$$

$$\phi_{\mathtt{E}4} = \mathtt{E}, \mathtt{!Connect}, \mathtt{A}[41,\infty], \mapsto \{(\mathtt{E}, \mathtt{?Hello}, \mathtt{G})\} \rhd [41,\infty]$$

$$\phi_{\mathtt{E}5} = \begin{matrix}\mathtt{E}, \mathtt{!Connect}, \mathtt{A}[0,\infty], \\ \mathtt{E}, \mathtt{?Hello}, \mathtt{G}[0,\infty]\end{matrix} \mapsto \left\{\begin{matrix}(\mathtt{E}, \mathtt{!Tea}, \mathtt{M}) \\ (\mathtt{E}, \mathtt{!Coffee}, \mathtt{M})\end{matrix}\right\} \rhd [0,\infty]$$

Fig. 4. Local invariants suite for web service E.

web service E. Let us denote that following the same pattern presented for E, it is easy to produce the invariants suites for the rest of web services.

The first invariant, by means $\phi_{\mathtt{E}1}$, means that on the one hand always that E receives Connect from A, followed by a time value less than or equal to 39 time units, then after sending Hello to the web service G, it will always receive Coffee from M; and on the other hand, that the sum of all time values, from Connect to Coffee is included in $[0, \infty]$.

Another example is the invariant $\phi_{\mathtt{E}2}$. It computes on the one hand that always that E receives Connect from A, followed by a time value less than or equal to 39 time units, then it will send Hello to the web service G, and on the other hand, that the sum of time values from Connect, to Hello belongs to [0,39].

Let us remark that just in the case of $\phi_{\mathtt{E}5}$, there are more than one item in the last set. This mean that the web service is allowed to perform any of these actions after matching its body. □

3.2 Global Invariants

Taking into account the monitoring structure presented in Figure **??**, usually we cannot assume that we have access to the global log. However, we would like to represent some properties involving more than one web service. We call a global log, as a log recorded in a centralized web service, where all actions are marked with the same time-stamp clock, thus there are not measure errors. In this case choreographies help us to check the correctness of these logs, due to the fact that they have a global clock and we can define properties as "local invariants" for

M.LOG= ⟨ A.!Connect,41 C.?Good Morning, 36.1, E.?Tea, ⟩
E.LOG= ⟨ A.!Connect,40.2 G.?Hello, 36, M.!Tea, ⟩
C.LOG= ⟨ A.!Connect,39.8 M.!Good Morning, 50.3, G.?Thank you, ⟩
G.LOG= ⟨ A.!Connect, 39.9 E.!Hello, 50.2, C.!Thank you, ⟩

Fig. 5. Set of local logs recorded in the web services M, E, C and G.

the choreography. In our approach we do not consider to have this global log, thus we introduce the notion of *global* invariants, which will help us to represent properties that involve many isolated local logs.

Let us present with our running example, an error produced in local logs that cannot be detected with local invariants. Let us consider the logs presented in the Figure **??**. As we can observe, all of them start with the synchronization input Connect from the web service A. After 40 time units the web services M and E send Good Morning to C and Hello to G(locally 41 time units in M and 40.2 time units in E). It could be possible that the clocks of the web services C, and G work slower than the one presented in M and E; thus, they receive this inputs on 39.8 and on 39.9. After that the web service M communicates with the web service E and the web service C with respect to the web service G. As we can observe, taking into account the choreography of our web services, presented in Figure **??**. This is not a correct situation of the local logs. The idea is that some web services perform the actions from s_6 to s_8, and the others perform the actions from s_6 to s_{10}. But, as we do not have a global log, only by checking the local behaviours we are unable to see that the set of logs represents an incompatible state.

To solve this problem, first we will define a *global log*, just adding one local log after another local log. The idea is to be able to represent properties over this global log, which help us to detect this kind of errors.

Definition 9. Let $\sigma_1 = \langle \vartheta_1^1, \ldots, \vartheta_n^1 \rangle, \ldots, \sigma_j = \langle \vartheta_1^j, \ldots, \vartheta_m^j \rangle$ be j local logs recorded from j different services. We will define a *global log* as the concatenation of these logs, that is $\sigma = \langle \vartheta_1^1, \ldots, \vartheta_n^1, 0, \ldots, 0, \vartheta_1^j, \ldots, \vartheta_m^j \rangle$. We let $\pi_{id}(\sigma)$ denote the projection of σ on a web service identifier id.

Let $\vartheta = (id'_a, \alpha, id)$ be an item of a visible trace, and id_a be a web service identifier. We define the following boolean function: $\mathtt{cmp}(\vartheta, id_a) = (id'_a = id_a)$. Let $\sigma = \langle \vartheta_1, \ldots, \vartheta_n \rangle$ be a global log, and id_a be a web service identifier. Formally, the projection is defined as:

$$\pi_{id_a}(\sigma) = \begin{cases} \langle\rangle & \text{if } n = 1 \wedge \neg \mathtt{cmp}(\vartheta_1, id_a) \\ \langle \vartheta_1 \rangle & \text{if } n = 1 \wedge \mathtt{cmp}(\vartheta_1, id_a) \\ \pi_{id_a}(\langle \vartheta_2, \ldots, \vartheta_n \rangle) & \text{if } \neg \mathtt{cmp}(\vartheta_1, id_a) \\ \langle \vartheta_1, t_1, \rangle \pi_{id_a}(\langle \vartheta_2, \ldots, \vartheta_n \rangle) & \text{if } \mathtt{cmp}(\vartheta_1, id_a) \end{cases}$$

Given two normalized visible traces σ_1, σ_2, we write $\sigma_1 \sim \sigma_2$ if σ_1 and σ_2 cannot be distinguished when making local observations, that is, we have that $\pi_{id}(\sigma_1) = \pi_{id}(\sigma_2)$ for all $id \in ID$. □

In our framework we assume that testers can combine the set of local logs taking into account any criterion. It is easy to proof that any two of them σ_i and σ_j are $\sigma_i \sim \sigma_j$. Following, we will define the notion of global invariants, and the correctness of global logs with respect to them.

Definition 10. Let $\mathcal{D} = (\mathcal{S}, \Sigma, ID, s_0, \{x\}, \mathcal{Z}, \mathcal{T})$ be a choreography. We say that the sequence ψ is a *global invariant* for $\mathcal{D}$, where ψ is defined according to the following EBNF:

$$\psi ::= SET_1 \mapsto_h SET_2$$

In this expression we consider that $h \in \{(1,1),(1,+),(+,+),(+,1)\}$, and $SET_1,\ SET_2 \subseteq \Phi_{ID}$. Let $\psi = SET_1 \mapsto_h SET_2$ be a global invariant, we will define the functions $\texttt{SET}_1(\psi) = SET_1$ and $\texttt{SET}_2(\psi) = SET_2$. Let σ be a global log. We will formally define the semantic of Correct, Incorrect (C/I) or Inconclusive:

h	Verdict	Condition
$\{1,1\}$	C/I	$\exists \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha \rightarrow \exists \phi_\beta \in \texttt{SET}_2(\psi) : \pi_\beta(\sigma) \diamond \phi_\beta$
$\{1,1\}$	Inconclusive	$\nexists \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha$
$\{1,+\}$	C/I	$\exists \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha \rightarrow \forall \phi_\beta \in \texttt{SET}_2(\psi) : \pi_\beta(\sigma) \diamond \phi_\beta$
$\{1,+\}$	Inconclusive	$\nexists \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha$
$\{+,1\}$	C/I	$\forall \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha \rightarrow \exists \phi_\beta \in \texttt{SET}_2(\psi) : \pi_\beta(\sigma) \diamond \phi_\beta$
$\{+,1\}$	Inconclusive	$\exists \phi_\alpha \in \texttt{SET}_1(\psi)$ such that $\pi_\alpha(\sigma) \neg \diamond \phi_\alpha$
$\{+,+\}$	C/I	$\forall \phi_\alpha \in \texttt{SET}_1(\psi) : \pi_\alpha(\sigma) \diamond \phi_\alpha \rightarrow \forall \phi_\beta \in \texttt{SET}_2(\psi) : \pi_\beta(\sigma) \diamond \phi_\beta$
$\{+,+\}$	Inconclusive	$\exists \phi_\alpha \in \texttt{SET}_1(\psi)$ such that $\pi_\alpha(\sigma) \neg \diamond \phi_\alpha$

The symbol $\diamond$ used to represent the correctness is overloaded, we will denote by $\sigma \diamond \psi$ that σ is *correct* with respect to ψ. □

Next, we define the correctness of global invariants with respect to both, the orchestration models and the choreography model. Let us remark that we are not allowed with any global clock for checking the temporal restrictions with respect to the choreography. So, on the one hand, we might check the correctness of the time restrictions of the global invariant against the web services and, on the other hand, we might check that the global invariant does not contradict the choreography.

Definition 11. Let $\psi = SET_1 \mapsto_h SET_2$ be a global invariant, and $\mathcal{D} = (\mathcal{S}, \Sigma, ID, s_0, \{x\}, \mathcal{Z}, \mathcal{T})$ be a choreography. We say that ψ is correct if for all $tr \in \texttt{NT}(D)$ we have that:

- Or $tr \diamond \psi$ or tr is inconclusive with respect to ψ, and there exists at least one tr that $tr \diamond \psi$.
- For all $\phi_\alpha \in SET_1 \cup SET_2$ we have that ϕ_α is correct with respect to α.

□

We conclude this section with the proposed problem of our running example. Let us summarize all information that we have, present the problem, and show the solution. We are provided with a set of web services, defined in Figure **??**,

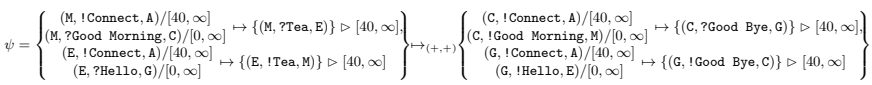

Fig. 6. Choreography of C, E, G and M.

which are modeled by using an adaptation of timed automata. For this set of web services, we have defined a choreography, also modeled by using a timed automata, and presented in Figure **??**. We have introduced a set of local invariants to check the correctness of the web services, see Figure **??**. Due to the fact we do not have a global clock, we cannot assume that all internal clocks of the web services work properly. Thus, we could have that some of them work faster than the others. This situation could produce the set of local logs presented in Figure **??**. As we discussed, with only local invariants we are not allowed to decide that this set of logs is incorrect. With the use of global invariants, we can detect this fault. Let us consider the global invariant presented in Figure **??**. When we check the correctness of the set of logs of the Figure **??** with respect to this invariant, we detect an error on them. Thus, we are able to detect an unexpected behaviour in the composition of these web services.

4 Conclusions and Future Work

In this work we have presented a formal framework to perform passive testing of distributed systems taking into account time information. We assume that we are provided with a formal specification of both the web services, and the interaction between them. These specifications are modeled by an adaptation of the well known timed automaton model. One contribution of this paper is to define (local)*invariants* for web services. A (local) invariant represents testing-properties, expresses by using input and outputs actions, for checking the correctness of the recorded traces (i.e, logs) of the system.

Another contribution of this paper is to discuss about some errors that are never detected only using local invariants. These errors are based into the idea that each web service has got its own set of local clock, and they do not share these clocks, it means, some of them can be faster that the others. Regarding, in our work we assume that we are not provided with a global clock. This scenario can produce erroneous undetectable situations using only sets of local invariants.

The last contribution of this paper is to provide a way to define (global) *invariants* for a set of web services without having a global clock. To finalize, we discuss that these invariants allow us to detect class of errors that we were unable to detect only using local invariants. As future work, we would like to upgrade our PASsive TEsting tool[1] with this methodology. On the one hand, we will implement the algorithms of checking the correctness of local and global invariants with respect to web services and choreography; and on the other hand

[1] (https://kimba.mat.ucm.es/paste)

algorithms for checking the correctness of logs with respect to local and global invariants.

Acknowledgments

We would like to thank the reviewers of this paper for the careful reading. The quality of the paper has notably increased by considering their useful comments and suggestions.

References

1. Eviware - soapUI: the Web Services Testing tool, 2010.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. C. Andrés, M.E. Cambronero, and M. Núñez. Formal passive testing of service-oriented systems. In *7th Int. Conf. on Services Computing, SCC'10*. IEEE Computer Society Press (in press), 2010. This paper is available at: http://kimba.mat.ucm.es/cesar/sccv4pp.pdf.
4. C. Andrés, M.G. Merayo, and M. Núñez. Passive testing of timed systems. In *6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311*, pages 418–427. Springer, 2008.
5. C. Andrés, M.G. Merayo, and M. Núñez. Supporting the extraction of timed properties for passive testing by using probabilistic user models. In *9th Int. Conf. on Quality Software, QSIC'09*, pages 145–154. IEEE Computer Society Press, 2009.
6. C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini. Towards automated wsdl-based testing of web services. In *6th Int. Conf. on Service-Oriented Computing, ICSOC'08, LNCS 5634*, pages 524–529. Springer, 2008.
7. C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini. WS-TAXI: a WSDL-based testing tool for web services. In *2nd Int. Conf. on Software Testing, Verification, and Validation, ICST'09*, pages 326–335. IEEE Computer Society Press, 2009.
8. E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi. A passive testing approach based on invariants: Application to the WAP. *Computer Networks*, 48(2):247–266, 2005.
9. A. Benharref, R. Dssouli, M. Serhani, and R. Glitho. Efficient traces' collection mechanisms for passive testing of web services. *Information & Software Technology*, 51(2):362–374, 2009.
10. L. Bentakouk, P. Poizat, and F. Zaïdi. A formal framework for service orchestration testing based on symbolic transition systems. In *Joint 21st IFIP TC6/WG6.1 Int. Conf. on Testing of Software and Communicating Systems, TestCom'09, and 9th Int. Workshop on Formal Approaches to Software Testing, FATES'09, LNCS 5826*, pages 16–32, 2009.
11. G. Díaz and I. Rodríguez. Automatically deriving choreography-conforming systems of services. In *6th IEEE Int. Conf. on Services Computing, SCC'09*, pages 9–16. IEEE Computer Society Press, 2009.
12. L. Frantzen, M. Las Nieves Huerta, Z.G. Kiss, and T. Wallet. On-the-fly model-based testing of web services with jambition. In *5th Int. Workshop on Web Services and Formal Methods, WS-FM'08, LNCS 5387*, pages 143–157. Springer, 2008.
13. M.G. Merayo, M. Núñez, and I. Rodríguez. Formal testing from timed finite state machines. *Computer Networks*, 52(2):432–460, 2008.

14. F. Raimondi, J. Skene, and W. Emmerich. Efficient online monitoring of web-service SLAs. In *16th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering, FSE'08*, pages 170–180. ACM Press, 2008.
15. J. Skene, A. Skene, J. Crampton, and W. Emmerich. The monitorability of service-level agreements for application-service provision. In *6th Int. Workshop on Software and Performance, WOSP'07*, pages 3–14. ACM Press, 2007.
16. V. Valero, M.E. Cambronero, G. Díaz, and H. Macià. A Petri net approach for the design and analysis of web services choreographies. *Journal of Logic and Algebraic Programming*, 78(6):359–380, 2009.
17. Y. Wang, M.Ü. Uyar, S.S. Batth, and M.A. Fecko. Fault masking by multiple timing faults in timed EFSM models. *Computer Networks*, 53(5):596–612, 2009.
18. S. Weerawarana, R. Chinnici, M. Gudgin, F. Curbera, and G. Meredith. Web services description language (WSDL), 2004. Version 2.0, 1.