Formal Passive Testing of Service-Oriented Systems

César Andrés*, M. Emilia Cambronero†, Manuel Núñez*
* Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
e-mails: c.andres@fdi.ucm.es, mn@sip.ucm.es
† Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha, Spain
e-mail: emicp@dsi.uclm.es

***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 behaviour 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.**

***Index Terms*—Passive Testing, Service Oriented Systems, Monitoring**

I. 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*.

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 she has to provide a degree of confidence of the system, only taking into account the monitored traces.

Research partially supported by the project TIN2009-14312-C02 and the UCM-BSCH programme to fund research groups (GR58/08 group number 910606).

In this paper we present a formal passive testing methodology to test web services. 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 [1], a timed extension of the framework defined in [2], to define invariants, and in [3] to define web service and choreography behaviours. In this paper, a Service-Oriented Scenario consists of a large number of services that interact with each other. For example, let **C**, **E**, **G**, and **M** be four web services that are interacting. When this set of services is monitored according to [4], we have the scheme presented in Figure 1. 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 **C**, **ISP**, **E**, **G**, and **M** 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 the 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. 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 [5], but there are some relevant differences. In [5] 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 [6]. 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.

The rest of this paper is structured as follows. First, in Section II is presented the preliminaries. Next, in Section III

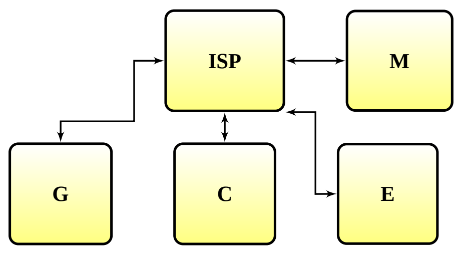

Figure 1. Structure of the Monitoring.

are introduced the local and global invariants. Finally, in Section IV we present our conclusions and some lines for future work.

II. Preliminaries

In this section we present our formalism to define web services and choreography models. We follow the ideas underlying the definition of web service and choreography model behaviours presented in [3]. However, instead of Finite State Machines, we use *Label Transition Systems* to represent web services. In the extended version of this paper we present this methodology by adding time information, using Timed Automata [7]. In our framework, we define the internal behaviour of a *web service* in terms of its interaction with other web services.

Example 1: Let us consider the web services **M**, **C**, **G**, and **E** presented in the Figure 2. For example, the transition on service **E**, $s_1 \xrightarrow{\mathbf{G}.?\texttt{START}} s_2$ represents that if **E** is in the state s_1, and it sends `START` to **G** then it will change its internal state to s_2.

If we consider the transition $s_1 \xrightarrow{\mathbf{M}.!\texttt{50 COINS}} s_2$, of the web service **C**, means that if **C** is in the state s_1 and it receives from **M** the message `50 COINS` then it will move to s_2. □

Next, we introduce the notion of *log* of a web service (aka a trace of a web service). As usual, a log is a sequence of *visible actions* of the web services. We denote by visible, all possible collected actions from the system.

Example 2: Regarding our previous example, we have that the set: {`START` , `CHECK` , `NOT CHECKED` , `50 COINS` , `10 COINS` , `20 COINS`} is the set of visible actions of the web services **M**, **C**, **G**, and **E**. In Figure 3 are represented four possible logs of these web services. For example, with **M**.LOG=⟨ **C**.?`50 COINS` , **E**.?`CHECK`⟩ we represent the log recorded on service **M**. This log contains two visible actions **C**.?`50 COINS` and **E**.?`CHECK`. The first one represents that **M** has performed `50 COINS` to **C**, meanwhile the second one represents that **M** has performed `CHECK` to **E**. □

Next we introduce our formalism to represent *choreographies*. Contrarily to systems of web services, 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. Concerning choreography machines, transitions are tuples (s, id, α, id', s') where s, s' are the initial and final states, α is the message, and id, id' are the sender and the addressee of the message, respectively.

Example 3: Let us consider the choreography presented in Figure 2. There is represented the global interaction of our four web services. For example, the transition $s_1\mathbf{E} \xrightarrow{\texttt{START}} \mathbf{G}s_2$ means that at the beginning, the service **E** has to send to **G** the message `START` and it will move to s_2, meanwhile the second transition, by means $s_2\mathbf{M} \xrightarrow{\texttt{50 COINS}} \mathbf{C}s_3$ denotes that the service **M** has to send to **C** the message `50 COINS`. □

III. Invariants

Next 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 behaviour 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.

The rest of the section is structure as follows. In Section III-A we review properties of isolated web services, and the local invariants will be defined, and in Section III-B we introduce the combination of isolated logs to check some significant properties, and the global invariants will be presented.

A. Local Invariants

Definition 1: Let Σ be a set of visible actions for a set of web services $\mathcal{A}$. We say that the sequence ϕ is a *local invariant* for the web service $id_a \in \mathcal{A}$, if it is defined according to the following EBNF:

$$\begin{array}{rcl} \phi_{id_a} & ::= & \texttt{Body} \mapsto \texttt{Head} \\ \texttt{Body} & ::= & id, \alpha, \texttt{Body} \mid \star, \texttt{Body'} \mid id, \alpha' \\ \texttt{Body'} & ::= & id, \alpha', \texttt{Body} \\ \texttt{Head} & ::= & \mathcal{O} \end{array}$$

In this expression we consider $\alpha' \in \Sigma$, $\alpha \in \Sigma \cup \{?\}$, $id \in \mathcal{A}$, and $\mathcal{O} \subseteq \mathcal{A} \times \Sigma$. The set of all local invariants for a set of web services $\mathcal{A}$ is denoted by $\Phi_{\mathcal{A}}$. □

Intuitively, the EBNF expresses that a local invariant is either a sequence of symbols where each component, but the last one, is either an expression id, α, with id being a web service identifier and with α being an action or the wildcard character $?$. There are two restrictions to this rule. First, a local invariant cannot contain two consecutive components $\star$ and $\star$ since this situation could be simulated by means of the expression $\star$. The second restriction is that an invariant cannot present a component of the form $\star$ followed by a wildcard character $?$, that is, the action of the next component must belong to Σ.

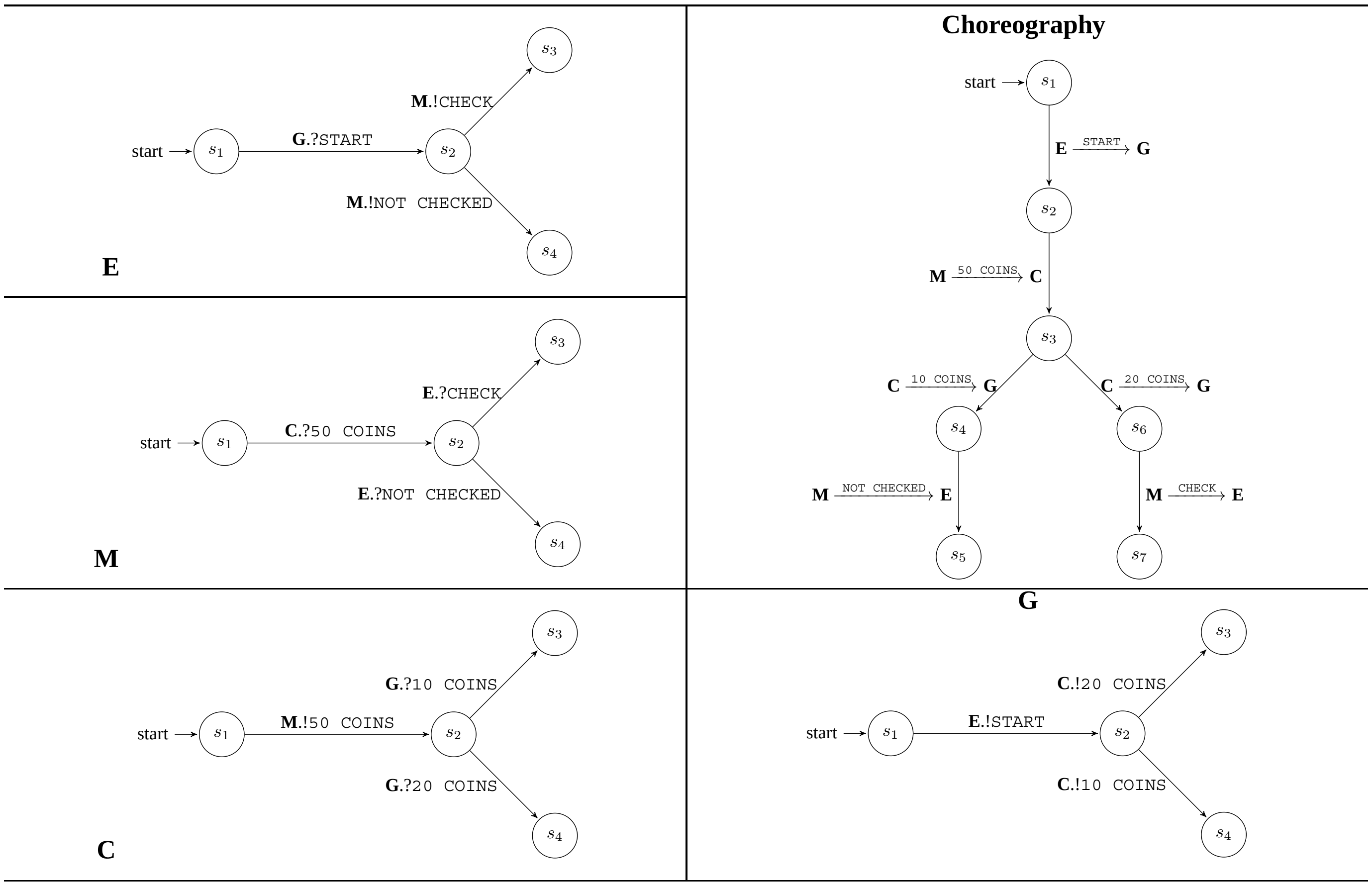

Figure 2. Web services **C**, **E**, **G** and **M** and its Choreography.

The last component, corresponding to the expression $id, \alpha, \mapsto \mathcal{O}$, is composed of a web service identifier associated with an message, by a set of tuples identifier/message. Let us remind that in our framework, the symbol ? can replace any message while the symbol $\star$ can replace a sequence of messages not containing the first message that appears in the part of the invariant that follows it.

Example 4: Next, we will illustrate the semantics of local invariants by using some examples for the web services presented in Figure 2.

$\phi_{\mathbf{M}}$ Always that **M** emits to **C** the message `50 COINS`, then **M** has to emit to **E** the message `CHECK` or the message `NOT CHECKED`. This is formally represented by:

$$\phi_{\mathbf{M}} = \mathbf{C}.?\texttt{50 COINS} \mapsto \left\{ \begin{array}{c} \mathbf{E}.?\texttt{CHECK}, \\ \mathbf{E}.?\texttt{NOT CHECKED} \end{array} \right\}$$

$\phi_{\mathbf{E}}$ Always that **E** emits to **G** the message `START`, then **M** will send to **E** the message `CHECK` or `NOT CHECKED`. This is formally represented by:

$$\phi_{\mathbf{E}} = \mathbf{G}.?\texttt{START} \mapsto \left\{ \begin{array}{c} \mathbf{M}.!\texttt{CHECK}, \\ \mathbf{M}.!\texttt{NOT CHECKED} \end{array} \right\}$$

$\phi_{\mathbf{C}}$ Always that **M** sends the message `50 COINS` to **C**, then **C** will send to **G** the message `10 COINS` or `20 COINS`. This is formally represented by:

$$\phi_{\mathbf{C}} = \mathbf{M}.!\texttt{50 COINS} \mapsto \left\{ \begin{array}{c} \mathbf{G}.?\texttt{10 COINS}, \\ \mathbf{G}.?\texttt{20 COINS} \end{array} \right\}$$

$\phi_{\mathbf{G}}$ Always that **E** sends the message `START` to **G** then **C** has to send the message `10 COINS` or `20 COINS` to **G**. This is formally represented by:

$$\phi_{\mathbf{G}} = \mathbf{E}.!\texttt{START} \mapsto \left\{ \begin{array}{c} \mathbf{C}.!\texttt{10 COINS}, \\ \mathbf{C}.!\texttt{20 COINS} \end{array} \right\}$$

Let us note that this set of invariants does not contradict the web services label transitions systems presented in Figure 2. □

When we check the correctness of a log with respect to an invariant, first we test if the log *matches* the `Body` of the invariant. When we find a sequence that matches, then we check the correctness of this sequence with respect to the `Consequent`. That is, let $\omega = \omega_1 \cdot (id, m)$ be the log of a system, and $\phi = \texttt{Body}_1 \mapsto \texttt{Head}_1$ be a local invariant. If we observe that ω_1 matches with $\texttt{Body}_1$ then (id, m) must be included in $\texttt{Head}_1$. 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.

Example 5: Let us consider the invariant $\phi_{\mathbf{C}}$ presented in Example 4. And the local log **C**.LOG presented in Figure 3.

$$\begin{array}{rlllr}
\mathbf{M}.\mathrm{LOG}= & \langle & \mathbf{C}.?\texttt{50 COINS}, & \mathbf{E}.?\texttt{CHECK} & \rangle \\
\mathbf{E}.\mathrm{LOG}= & \langle & \mathbf{G}.?\texttt{START}, & \mathbf{M}.!\texttt{CHECK} & \rangle \\
\mathbf{C}.\mathrm{LOG}= & \langle & \mathbf{M}.!\texttt{50 COINS}, & \mathbf{G}.?\texttt{10 COINS} & \rangle \\
\mathbf{G}.\mathrm{LOG}= & \langle & \mathbf{E}.!\texttt{START}, & \mathbf{C}.!\texttt{10 COINS} & \rangle
\end{array}$$

Figure 3. Local logs recorded on web services **C**, **E**, **G** and **M**

We have that **C**.LOG can be checked by $\phi_{\mathbf{C}}$ because **M**.!50 COINS matches, and the result of checking this log with respect to the invariant is correct due to the fact that **G**.?10 COINS$\in \{$**G**.?10 COINS ,**G**.?20 COINS$\}$. Let us note the set of all logs presented in Figure 3 is correct with respect to the invariants presented in Example 4. □

B. Global Invariants

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. So, we introduce the notion of *global* invariants, which will help us to represent properties that involve many isolated local logs. We will use the notions of choreography and web services in order to check the correctness of global invariants.

Definition 2: Let $\mathcal{A}$ be a set of web services, and Σ be a set of visible actions. We say that the sequence ψ is a *global invariant* for $\mathcal{A}$, if it is defined according to the following EBNF:

$$\psi ::= \quad \texttt{SET}_1 \mapsto_h \texttt{SET}_2$$

In this expression we consider $\texttt{SET}_1$ and $\texttt{SET}_2 \subseteq \Phi_{\mathcal{A}}$ and $h \in \{(1,1),(1,+),(+,+),(+,1)\}$. □

Intuitively, the previous EBNF represents how to combine set of local logs, with respect to some operations. Let us consider the global invariant $\texttt{SET}_1 \mapsto_{(1,1)} \texttt{SET}_2$. It represents that if we have a local invariant for the web service id, presented in $\texttt{SET}_1$ that return correct when it checks the correctness of the local log of id, then there must be at least one local invariant presented in $\texttt{SET}_2$ that will return true when it checks the correctness of the log. The use of the character $+$ implies the set of all invariants. That is, the meaning of $\texttt{SET}_1 \mapsto_{(+,1)} \texttt{SET}_2$ represents that if we have that the set of local invariants $\texttt{SET}_1$ return correct with respect to their local observations, then we have to find a local log that is correct with respect to a local invariant presented in $\texttt{SET}_2$.

With respect to the correctness of the global invariants, we have on the one hand that the local invariants belonging to $\texttt{SET}_1$ and $\texttt{SET}_2$ may be correct or not with respect to the web service, but at least one time they have to return true with a correct log. On the other hand, regarding the global property represented by these two sets, taking into account the chosen $h-$operator, it must be conform with respect to the choreography. That is, the global invariant does not contradict the choreography.

Example 6: Next, we will illustrate the semantics of a global invariant by using an example for the web services presented in Figure 2. Let us consider the following property: *"Always that **M** sends both 50 COINS to **C** and CHECK to **E** then we will get that **C** also has sent to **G** the message 20 COINS"*. This is formally represented by:

$$\left\{\begin{array}{c}\phi_{\mathbf{M}} = \mathbf{C}.?\texttt{50 COINS} \mapsto \{\mathbf{E}.?\texttt{CHECK}\},\\ \phi_{\mathbf{E}} = \mathbf{G}.?\texttt{START} \mapsto \{\mathbf{M}.!\texttt{CHECK}\}\end{array}\right\} \mapsto_{(+,+)} \left\{\begin{array}{c}\phi_{\mathbf{C}} = \mathbf{M}.!\texttt{50 COINS} \mapsto \{\mathbf{G}.?\texttt{20 COINS}\},\\ \phi_{\mathbf{G}} = \mathbf{E}.!\texttt{START} \mapsto \{\mathbf{C}.!\texttt{20 COINS}\}\end{array}\right\}$$

□

We conclude the paper showing that global invariants can detect errors that cannot be found by using only local invariants. Let us consider the web services choreography presented in Figure 2 and the logs presented in Figure 3, recorded from the web services. From a local point of view, these logs can be recorded on web services **M**, **C**, **E**, and **G**; it means, there does not exists any correct local invariant which these logs contradict. And, we can easily check that these logs cannot be performed from the choreography presented in Figure 2. So, if we consider the global invariant presented in Example 6 we are allowed to detect an error on these logs. The use of global invariants is powerful, but more *expensive* (in computation therms) than checking the correctness of local invariants, so some properties must be checked on local invariants and those which are unable to check, would be represented by global invariants.

IV. Conclusions and Future Work

In this work we have presented a new methodology to represent properties of web services. These properties, called invariants, are used within a passive testing methodology. A set of invariants represents the most relevant expected behaviour of the system. In order to increase the applicability of our methodology, we assume that web services can be monitored in their environments, but we are not allowed to monitor all the interaction among them. Therefore, our invariants can represent properties that involve different web services, based on their local observations. As future work, we would like to introduce time information, and data information into invariants to be able to represent more complex properties.

References

[1] C. Andrés, M. Merayo, and M. Núñez, "Passive testing of timed systems," in *6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311*, pp. 418–427, Springer, 2008.

[2] E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi, "A passive testing approach based on invariants: Application to the WAP," *Computer Networks*, vol. 48, no. 2, pp. 247–266, 2005.

[3] G. Díaz and I. Rodríguez, "Automatically deriving choreography-conforming systems of services," in *6th IEEE Int. Conf. on Services Computing, SCC'09*, pp. 9–16, IEEE Computer Society Press, 2009.

[4] 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*, pp. 3–14, ACM Press, 2007.

[5] 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*, pp. 170–180, ACM Press, 2008.

[6] C. Andrés, M. 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*, pp. 145–154, IEEE Computer Society Press, 2009.

[7] C. Andrés, M. Cambronero, and M. Núñez, "Formal passive testing of service-oriented systems: Extended version," 2010. Available at: http://kimba.mat.ucm.es/cesar/wspassive.pdf.