Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 48 (2005) 247–266
Computer Networks
www.elsevier.com/locate/comnet

A passive testing approach based on invariants: application to the WAP ☆

Emmanuel Bayse [a], Ana Cavalli [a,*], Manuel Núñez [b], Fatiha Zaïdi [a]

[a] *Institut National des Télécommunications, GET-INT, LOR, 9 rue Charles Fourier, 91011 Evry cedex, France*
[b] *Dept. Sistemas Informáticos y Programación, Universidad Complutense de Madrid, E-28040 Madrid, Spain*

Received 6 November 2003; received in revised form 11 May 2004; accepted 8 September 2004
Available online 12 January 2005

Responsible Editor: R. Gotzhein

Abstract

This paper presents a new methodology to perform passive testing based on invariants. This novel approach is supported by the following idea: a set of invariants represent the most relevant expected properties of the implementation under test. Intuitively, an invariant expresses the fact that each time the implementation under test performs a given sequence of actions, then it must exhibit a behavior reflected in the invariant. For example, an invariant such as $i_1/o_1,\ldots,i_{n-1}/o_{n-1},i_n/O$ must be interpreted as "*each time the implementation performs the sequence* $i_1/o_1,\ldots,i_{n-1}/o_{n-1},i_n$ *the next observed output belongs to the set* O". We call these invariants *simple invariants*. In this work we introduce a new notion of invariants to deal with more subtle properties. For instance, we will consider invariants to express properties such as "*if y happens then we must have that x had happened before*". These invariants are called *obligation invariants*. We present algorithms to decide the correctness of the proposed invariants with respect to a given specification. Once we have that an invariant is correct with respect to a given specification, we check whether the execution traces observed from the implementation *respect* the invariant. In order to perform this phase we present two algorithms based, respectively, on left-to-right and right-to-left pattern matching algorithms.

In addition to the theoretical framework we have developed a software tool, called TestInv, that helps in the automation of our passive testing approach. In particular, the algorithms presented in this paper are fully implemented in

☆ Research supported in part by the Spanish *Ministerio de Ciencia y Tecnología* MCyT project *MASTER* (TIC2003-07848-C02-01), the Junta de Castilla-La Mancha project *DISMEF* (PAC-03-001) and the Marie Curie RTN *TAROT* (MCRTN 05121). This research was carried out while the third author was visiting the GET-INT under the financial support of the Platonis project.

* Corresponding author. Tel.: +33 1 60 76 44 27; fax: +33 1 60 76 47 11.
E-mail addresses: emmanuel.bayse@int-evry.fr (E. Bayse), ana.cavalli@int-evry.fr (A. Cavalli), mn@sip.ucm.es (M. Núñez), fatiha.zaidi@int-evry.fr (F. Zaïdi).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2004.09.009

the tool. Finally, in order to test the usefulness of our approach we have chosen a real-life case study: the Wireless Application Protocol (WAP). We present a test architecture as well as the most relevant results obtained from the application of our approach to the WAP.

Keywords: Passive testing; Conformance testing; Invariants; Software tools for testing; Wireless Application Protocol (WAP)

1. Introduction

The activity of conformance testing is essentially focused on verifying the conformity of a given implementation to its specification. In most cases testing is based on the ability of a tester that stimulates the implementation under test and checks the correction of the answers provided by the implementation [10,13]. However, in some situations this activity becomes difficult and even impossible to perform. For example, this is the case if the tester is not provided with a direct interface to interact with the implementation under test (IUT). Another conflictive situation appears when the implementation is built from components that are running in their environment and cannot be shutdown or interrupted for a long period of time. In these situations, there is a particular interest in using other types of testing techniques such as *passive testing*. In passive testing the tester does not need to interact with the IUT. On the contrary, the execution traces are observed without interfering with the behavior of the IUT. Passive testing has very large domains of application. For instance, it can be used as a monitoring technique to detect and report errors (this is the use that we consider in this paper). Another area of application is in network management to detect configuration problems, fault identification, or resource provisioning (e.g. [14,21]). It can be also used to study the feasibility of new features as classes of services, network security, and congestion control.

Even though passive testing techniques are not new (see for example the approach shown in [1]) in the last years a very active research on passive testing has been developed. Usually, the execution traces of the implementation are compared with the specification to detect faults in the implementation [12,15,18,19]. In general, the specification has the form of a finite state machine (FSM) and the work consisted in verifying that the executed trace is accepted by the FSM specification. A drawback of these first approaches is the low performance of the proposed algorithms (in terms of complexity in the worst case) if non-deterministic specifications are considered.

A new approach was proposed in [6]. There, a set of properties, called *invariants*, were extracted from the specification and checked on the traces observed from the implementation to test their correctness. That is, in this approach information was extracted from the specification and then used to process the trace. However, one of the drawbacks of this work is the limitation on the grammar used to express invariants. For instance, properties as

Each time that a user asks for connection and the connection is granted, if after performing some operations the user asks for disconnection then he is disconnected.

could not be easily represented by using their invariants since all the possible sequences of actions expressing the idea of *some operations* must be explicitly written.

A new formalism to express invariants was presented in [2]. For instance, the possibility of specifying wild-card characters in invariants was added. Besides, a set of outputs was allowed (instead of a single output) as termination of the invariant. Thus, properties such as

Each time that a user asks for a resource (e.g. a web page) either the resource is obtained or an error is produced.

could be easily specified. However, heavy experimentation using the invariants approach reported in [2,6] has shown additional lines for improvement. For example, properties such as