Science of Computer Programming 122 (2016) 1–28

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Safe and efficient runtime testing framework applied in dynamic and distributed systems

Mariam Lahami [a,*], Moez Krichen [a,b], Mohamed Jmaiel [a,c]

[a] ReDCAD Laboratory, University of Sfax, National School of Engineers of Sfax, B.P. 1173, 3038 Sfax, Tunisia
[b] Faculty of Computer Science and Information Technology, Al-Baha University, Saudi Arabia
[c] Research Center for Computer Science, Multimedia and Digital Data Processing of Sfax, B.P. 275, Sakiet Ezzit, 3021 Sfax, Tunisia

ARTICLE INFO

Article history:
Received 26 November 2014
Received in revised form 2 February 2016
Accepted 5 February 2016
Available online 20 February 2016

Keywords:
Runtime testing
Dynamic structural adaptations
Distributed test execution
Resource awareness
TTCN-3

ABSTRACT

This paper provides a standard-based and resource aware **R**untime **T**esting **F**ramework For **A**daptable and **D**istributed **S**ystems (RTF4ADS). Based on the runtime testing approach, RTF4ADS performs safely and efficiently tests on the final operational environment of such dynamic systems. Indeed, our proposal (1) looks for a minimal set of tests to re-execute written in a standardized notation, (2) assigns the involved test components in execution nodes while respecting resources and connectivity constraints and finally (3) performs distributed testing at runtime while it prevents test processes from interfering with business processes. Implementation details of the proposed research prototype are presented. To demonstrate the validity and the usefulness of RTF4ADS, a case study in the healthcare domain implemented using the Open Services Gateway Initiative (OSGi) platform is illustrated. The experiments that we conducted show a reasonable overhead introduced by RTF4ADS. They also demonstrate the efficiency of this framework in testing dynamic and distributed systems while reducing side effects of this validation technique.

1. Introduction

Nowadays, critical and distributed component-based systems tend to evolve at runtime. In general, such evolution is required to provide more dependable systems, to remove identified deficiencies, or to handle the increased variability of the execution context (e.g., mobility of devices hosting components) and the rapid evolution of user requirements. Stopping the system during its execution, in order to perform an upgrade, may be often inconvenient and unsafe, notably for critical systems such as crisis management systems, remote medical care systems, etc. For this reason, runtime evolution, commonly referred to as dynamic adaptation, is more and more required. It is performed either by dynamically modifying the architecture of the software system (i.e., structural adaptations) or by modifying its behavior (i.e., behavioural adaptations).

Nevertheless, runtime adaptations of component-based systems may generate new risks of bugs, unpredicted interactions and performance degradation. This may cause system malfunctions and guide its execution to an unsafe state. For instance, a required functionality may be removed by mistake when removing a component or an undesired cycle may be introduced in new interactions between components. Therefore, guaranteeing the high quality and trustworthiness of dynamically adaptable systems is becoming a crucial requirement to be considered.

* Corresponding author.
E-mail address: mariam.lahami@redcad.org (M. Lahami).

http://dx.doi.org/10.1016/j.scico.2016.02.002
0167-6423/

As one of the key methods to get confidence in adaptable and distributed systems, runtime testing gained research focus during the last decade. It is defined in [1] as any testing method (i.e., unit testing, regression testing, conformance testing, etc.) that is carried out in the final execution environment during the operation time of a software system.

Several studies have considered runtime testing in various software domains [2–12]. Most of these approaches provide platform dependent test frameworks like JUnit[1] to specify and execute tests. Moreover, the test distribution over the network has been rarely addressed. Most of the studied works assume that tests are integrated into the components under test. Thus, managing test case assignment to the test components and also managing their deployment in host computers have been considered only by [7,13]. Furthermore, we notice that a major issue related to resource awareness during runtime testing has been neglected by the majority, except [3].

Despite the effort to apply effective runtime testing process, it remains one of the most challenging and expensive validation technique. For handling the above limitations, we design a test framework executing runtime tests in a cost effective manner. This proposal is called Runtime Testing Framework for Adaptable and Distributed Systems (RTF4ADS). Firstly, RTF4ADS ensures the online execution of a subset of test cases covering only software components or compositions affected by the dynamic change. Secondly, it is distinguishably characterized by its *resource awareness*. To this end, the assignment of test cases to execution nodes is done while fitting resource and connectivity constraints of the execution environment. Thirdly, it offers a *standard-based* and *platform-independent* test execution infrastructure based on the Testing and Test Control Notation Version 3 (TTCN-3) language [14]. The adopted TTCN-3 test architecture is extended with a test isolation module with the aim of preventing interference between test and business behaviors.

To show the relevance of RTF4ADS, a case study in the healthcare domain, called Teleservices and Remote Medical Care System (TRMCS), is used. This service oriented application is implemented using the Open Service Gateway Initiative (OSGi) platform [15]. In order to evaluate RTF4ADS, several experiments were conducted while varying the System Under Test (SUT) architecture and the execution environment topology. Results prove that the proposed framework can efficiently check the SUT correctness without disturbing its normal work at runtime. The assessed overhead in terms of execution time and memory consumption is relatively low and can be tolerated especially when dynamic adaptations are not frequently performed.

Compared to our previous publications [16–19], this paper presents a valuable contribution by gathering all these independent parts into a well-implemented prototype in which transitions from one step to another are fully automated. In fact, the current version of RTF4ADS is extended with test selection and test placement facilities. With respect to the test selection step, a new algorithm is proposed to identify affected compositions by different kinds of reconfiguration actions and then to look for their corresponding integration tests. About test component placement step, the assignment of test components to the execution nodes is improved by fitting a new constraint, namely link bandwidth capacities. Concerning test isolation and execution steps, the test isolation infrastructure is enhanced with a new aspect-based test isolation technique that improves the runtime testability of software components. In addition, the distributed test execution is realized for the first time in a real distributed environment and fully implemented and managed through a graphical user interface. At last, the overhead assessment of RTF4ADS is newly conducted through several experiments especially to stress the importance of resource awareness and test distribution to decrease its overhead as well as the utility of the test isolation layer to cope with safe runtime testing.

The remainder of this paper is organized as follows. Section 2 introduces main concepts related to runtime testing of adaptable and distributed systems. Background materials on TTCN-3 standard is outlined in Section 3. The different modules included in RTF4ADS are overviewed in Section 4 and their implementation details are given in Section 5. Section 6 presents its application on the TRMCS case study, followed by an overhead estimation. Some related work is discussed in Section 7. Finally, Section 8 concludes the paper and exposes some future work.

2. Background on runtime testing of dynamically adaptable and distributed component-based systems

In this section, we present the architecture of distributed software systems and the different adaptation actions that can be performed on these systems at runtime. Subsequently, a runtime testing approach is introduced and prerequisites for its safe application are discussed.

2.1. Dynamically adaptable and distributed component-based systems

A component is a software entity that encapsulates a set of functions or data. Seen as black-boxes, components offer functionalities that are expressed by clearly defined interfaces. These interfaces are usually required to connect components for communication and to compose them in order to provide complex functionalities. In other words, components are capable of exposing these functionalities as provided interfaces (called also services) to other components or using other functionalities from other components by their required interfaces.

Currently, software systems made up of such components are being implemented as distributed systems (see Fig. 1 where the architecture of a distributed software system is displayed). This issue is manifested by an execution environment

[1] http://www.junit.org/.