Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 329 (2016) 123–148

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Test Scenario Generation from Natural Language Requirements Descriptions based on Petri-Nets

Edgar Sarmiento [1,2] Julio C. S. P. Leite [3]

Departament of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil

Eduardo Almentero [4]

Mathematics Department
Universidade Federal Rural do Rio de Janeiro
Rio de Janeiro, Brazil

Guina Sotomayor Alzamora [5]

Instituto de Matemática y Ciencias Afines
Universidad Nacional de Ingeniería
Lima, Peru

Abstract

Test generation from functional requirements in natural language (NL) is often time-consuming and error prone, especially in complex projects. In this context, formal representations like Petri-Nets are increasingly used as input for automated test scenario generation. However, formal representations are not trivial, and it requires a strong knowledge on formal modeling. In this paper we propose an approach to generate test scenarios that takes as input a Restricted-form of Natural Language (RNL) requirements specification. This approach translates automatically RNL requirements specified as Scenarios into executable Petri-Net models; these Petri-Nets are used as input model for test scenario generation. Our approach checks the quality of the input models and aims to decrease the time and the effort with respect to test scenario generation process. Demonstration of the feasibility of the proposed approach is based on an example of use that describes the operation of the approach.

Keywords: scenario, requirements, petri-nets, testing, test scenario.

[1] Thanks to CAPES funding agency for financial support

[2] Email: ecalisaya@inf.puc-rio.br

[3] Email: julio@inf.puc-rio.br

[4] Email: almentero@ufrrj.br

[5] Email: guinas@gmail.com

http://dx.doi.org/10.1016/j.entcs.2016.12.008

1571-0661/© 2016 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1 Introduction

Software testing is one of the validation techniques most commonly used, this approach improves the quality of the final product, particularly checking that the software behavior meets its requirements. However, test generation and execution tasks are still quite expensive and usually done manually, then the automation of testing process is a challenging topic.

The Model-based Testing (MBT) is an alternative to the automation of these tasks, in which tests are derived from system specifications. Thus, the expected system behavior is described using formal specification notations [2]. MBT refers to black-box testing method in which test scenarios and oracle are automatically generated from a formal and functional model of a System Under Test (SUT). An important benefit of MBT, among other things, is automatically generate test scenarios from a model of a system under test and the automatically validate these test scenarios by executing the system under test and comparing their results against the expected results. The main shortcomings of MBT are the model construction and selection of suitable formal notations [24], often most of the proposed approaches require manual intervention or the creation of additional complex behavioral models. According to [25], this significantly hinders their applicability in practice.

In most of existing approaches for generating test scenarios for model-based systems, testing practitioners usually decompose the system in different use scenarios, then, formal representations are created for each identified scenario. The test scenarios are derived from these intermediate formal representations. According to [2], the quality of these specifications is crucial for an effective testing campaign; thus, it is desirable to describe the expected system behavior via some (semi-)formal notations. Examples of formal notation are Petri-Nets [17] or Communicating Sequential Processes (CSP) [19].

The use of (semi-)formal notations facilitates the process of test automation. However, this practice is expensive and not widely used in industrial practice. On the other hand, in order to allow for an easy communication between clients and developers, natural language-based representations are frequently used in Requirements Engineering. In this context, functional requirements are represented as scenarios and described by specific flows of events, which are based on user perspective. The use of scenarios helps understanding a specific situation in an application, prioritizing their behavior [14]. Some of the most prominent languages to write scenarios are restricted-form of use case descriptions [5], [9]; scenario representation [14]; UML dynamic behavior diagrams; and Message Sequence Charts [1]. Although some of these languages provide an accessible visualization of models, they lack formal semantics to support further analysis or test generation.

In this context, *scenario specifications* are usually informal or semi-formal, and due to natural language ambiguity, they cannot be used directly for MBT activities. In order to perform an automated MBT from these scenarios, it is necessary: (i) to detect and fix defects within scenarios; (ii) to translate them from informal to formal representations, like Petri-Nets; and (iii) to derive testing from formal

Download English Version:

https://daneshyari.com/en/article/4950045

Download Persian Version:

https://daneshyari.com/article/4950045

Daneshyari.com