Information and Software Technology xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Search-based automated testing of continuous controllers: Framework, tool support, and case studies

Reza Matinnejad [a,*], Shiva Nejati [a], Lionel Briand [a], Thomas Bruckmann [b], Claude Poull [b]

[a] SnT Center, University of Luxembourg, 4 rue Alphonse Weicker, L-2721 Luxembourg, Luxembourg
[b] Delphi Automotive Systems, Avenue de Luxembourg, L-4940 Bascharage, Luxembourg

ARTICLE INFO

Article history:
Received 29 November 2013
Received in revised form 29 April 2014
Accepted 10 May 2014
Available online xxxx

Keywords:
Search-based testing
Continuous controllers
Model-in-the-loop testing
Automotive software systems
Simulink models

ABSTRACT

Context: Testing and verification of automotive embedded software is a major challenge. Software production in automotive domain comprises three stages: Developing automotive functions as Simulink models, generating code from the models, and deploying the resulting code on hardware devices. Automotive software artifacts are subject to three rounds of testing corresponding to the three production stages: Model-in-the-Loop (MiL), Software-in-the-Loop (SiL) and Hardware-in-the-Loop (HiL) testing.
Objective: We study testing of continuous controllers at the Model-in-Loop (MiL) level where both the controller and the environment are represented by models and connected in a closed loop system. These controllers make up a large part of automotive functions, and monitor and control the operating conditions of physical devices.
Method: We identify a set of requirements characterizing the behavior of continuous controllers, and develop a search-based technique based on random search, adaptive random search, hill climbing and simulated annealing algorithms to automatically identify worst-case test scenarios which are utilized to generate test cases for these requirements.
Results: We evaluated our approach by applying it to an industrial automotive controller (with 443 Simulink blocks) and to a publicly available controller (with 21 Simulink blocks). Our experience shows that automatically generated test cases lead to MiL level simulations indicating potential violations of the system requirements. Further, not only does our approach generate significantly better test cases faster than random test case generation, but it also achieves better results than test scenarios devised by domain experts. Finally, our generated test cases uncover discrepancies between environment models and the real world when they are applied at the Hardware-in-the-Loop (HiL) level.
Conclusion: We propose an automated approach to MiL testing of continuous controllers using search. The approach is implemented in a tool and has been successfully applied to a real case study from the automotive domain.

1. Introduction

Modern vehicles are increasingly equipped with Electronic Control Units (ECUs). The amount and the complexity of software embedded in the ECUs of today's vehicles is rapidly increasing. To ensure the high quality of software and software-based functions on ECUs, the automotive and ECU manufacturers have to rely on effective techniques for verification and validation of their software systems. A large group of automotive software functions require to monitor and control the operating conditions of physical components. Examples are functions controlling engines, brakes, seatbelts, and airbags. These controllers are widely studied in the control theory domain as *continuous controllers* [1,2] where the focus has been to *optimize* their design for a specific application or a specific hardware configuration [3–5]. Yet a complementary and important problem, of how to systematically and automatically *test* controllers to ensure their correctness and safety, has received almost no attention in control engineering research [1].

In this article, we concentrate on the problem of automatic and systematic test case generation for continuous controllers. The principal challenges when analyzing such controllers stem from their continuous interactions with the physical environment, usually through feedback loops where the environment impacts computations and vice versa. We study the testing of controllers

* Corresponding author. Tel.: +352 621370479.
E-mail addresses: reza.matinnejad@uni.lu (R. Matinnejad), shiva.nejati@uni.lu (S. Nejati), lionel.briand@uni.lu (L. Briand), thomas.bruckmann@delphi.com (T. Bruckmann), claude.poull@delphi.com (C. Poull).

http://dx.doi.org/10.1016/j.infsof.2014.05.007
0950-5849/

at an early stage where both the controller and the environment are represented by models and connected in a closed feedback loop process. In model-based approaches to embedded software design, this level is referred to as *Model-in-the-Loop (MiL)* testing.

Testing continuous aspects of control systems is challenging and is not yet supported by existing tools and techniques [1,3,4]. There is a large body of research on testing *mixed* discrete–continuous behaviors of embedded software systems where the system under test is represented using state machines, hybrid automata, and hybrid petri nets [6–8]. For these models, test case generation techniques have been introduced based on meta-heuristic search and model checking [9,10]. These techniques, however, are not amenable to testing *purely* continuous properties of controllers described in terms of mathematical models, and in particular, differential equations. A number of commercial verification and testing tools have been developed, aiming to generate test cases for MATLAB/Simulink models, namely the Simulink Design Verifier software [11], and Reactis Tester [12]. Currently, these tools handle only combinatorial and logical blocks of the MATLAB/Simulink models, and fail to generate test cases that specifically evaluate continuous blocks (e.g., integrators) [1].

Contributions. In this article, we propose a search-based approach to automate generation of MiL level test cases for continuous controllers. This article extends and refines an earlier version of this work which appeared in [13], making the following contributions: We identify a set of common requirements characterizing the desired behavior of such controllers. We develop a search-based technique to generate stress test cases attempting to violate these requirements by combining *explorative* and *exploitative* search algorithms [14]. Specifically, we first apply a purely explorative random search to evaluate a number of input signals distributed across the search space. Combining the domain experts' knowledge and random search results, we select a number of regions that are more likely to lead to critical requirement violations in practice. We then start from the worst-case input signals found during exploration, and apply an exploitative *single-state* search [14] to the selected regions to identify test cases for the controller requirements. Our search algorithms rely on *objective* functions created by formalizing the controller requirements.

We have implemented our approach in a tool, called Continuous Controller Tester (CoCoTest). We evaluated our approach by applying it to an automotive air compressor module and to a publicly available controller model. Our experiments show that our approach automatically generates several test cases for which the MiL level simulations indicate potential errors in the controller model or in the environment model. Furthermore, the resulting test cases had not been previously found by manual testing based on domain expertise. In addition, our approach computes test cases better and faster than a random test case generation strategy. Finally, our generated test cases uncover discrepancies between environment models and the real world when they are applied at the Hardware-in-the-Loop (HiL) level.

Organization. This article is organized as follows. Section 2 introduces the industrial background of our research. Section 3 precisely formulates the problem we aim to address in this article. Section 4 outlines our solution approach and describes how we cast our MiL testing approach as a search problem. Our MiL testing tool, CoCoTest, and the results of our evaluation of the proposed MiL testing approach are presented in Sections 5 and 6, respectively. Section 7 compares our contributions with related work. Finally, Section 8 concludes the article.

2. MiL testing of continuous controllers: practice and challenges

Control system development involves building of control software (controllers) to interact with mechatronic systems usually referred to as plants or environment [2]. An abstract view of such controllers and their plant models is shown in Fig. 1(a). These controllers are commonly used in many domains such as manufacturing, robotics, and automotive. Model-based development of control systems is typically carried out in three major levels described below. The models created through these levels become increasingly more similar to real controllers, while verification and testing of these models becomes successively more complex and expensive.

Model-in-the-Loop (MiL): At this level, a model for the controller and a model for the plant are created in the same notation and connected in the same diagram. In many sectors and in particular in the automotive domain, these models are created in MATLAB/Simulink. The MiL simulation and testing is performed entirely in a virtual environment and without any need for any physical component. The focus of MiL testing is to verify the control behavior or logic, and to ensure that the interactions between the controller and the plant do not violate the system requirements.

Software-in-the-Loop (SiL): At this level, the controller model is converted to code (either autocoded or manually). This often includes the conversion of floating point data types into fixed-point values as well as addition of hardware-specific libraries. The testing at the SiL level is still performed in a virtual and simulated environment like MiL, but the focus is on controller code which can run on the target platform. Further, in contrast to verifying the behavior, SiL testing aims to ensure correctness of floating point to fixed-point conversion and the conformance of code to control models, especially in contexts where coding is (partly) manual.

Hardware-in-the-Loop (HiL): At this level, the controller software is fully installed into the final control system (e.g., in our case, the controller software is installed on the ECUs). The plant is either a real piece of hardware, or is some software (HiL plant model) that runs on a real-time computer with physical signal simulation to lead the controller into believing that it is being used on a real plant. The main objective of HiL is to verify the integration of hardware and software in a more realistic environment. HiL testing is the closest to reality, but is also the most expensive among the three testing levels and performing the test takes the longest at this level.

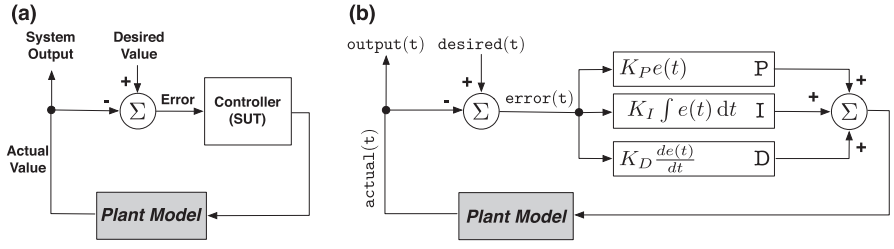

Fig. 1. Continuous controllers: (a) A MiL level controller-plant model, and (b) a generic PID formulation of a continuous controller.

Please cite this article in press as: R. Matinnejad et al., Search-based automated testing of continuous controllers: Framework, tool support, and case studies, Inform. Softw. Technol. (2014), http://dx.doi.org/10.1016/j.infsof.2014.05.007