International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-1 Issue-2, June 2012

Hardware -in-the-Loop Search –Based Testing Approach to Embedded Systems

Ashwini Motghare, Swapnili P. Karmore

Abstract:- The complexity of embedded systems is ever increasing while high system quality is being demanded at the same time. With the continuously growing software and system complexity in electronic control units and shortening release cycles, the need for efficient testing grows. In order to perform testing of electronic control units in practice search-based hardware-in-the-loop test environments are used to run the system under test in a simulation environment under real-time conditions. The potential of applying search-based testing approach to the functional testing has been demonstrated in various test cases. The focus was mainly on simulating the system under test in order to evaluate test cases. However, in many cases only the final hardware unit is available for testing. This paper present an approach in which evolutionary functional testing is performed using an actual electronic control unit for test case evaluation. An extensive case study has been carried out to access its capabilities. We demonstrate the use of evolutionary testing for functional testing in an industrial setting by applying the developed solution to test functioning of serial production of an automation-system electronic control unit.

Index Terms:- Evolutionary Algorithm (EA), Functional Testing, Hardware-in-the-loop-Testing (HiL), Automation-System (AS) .

I. INTRODUCTION

The complexity of embedded software systems is ever increasing while high software quality is being demanded at the same time. With the continuously growing software and system complexity in electronic control units and shortening release cycles, the need for efficient testing grows. Erroneous embedded systems may cause serious accidents related to human safety. Therefore it is essential to detect any errors in the embedded systems through various test processes. The likelihood of finding errors depends on the quality of test case design. In cases of manual test case design the test quality directly depends on the skill of the tester. If relevant test cases are forgotten, the probability of finding errors is reduced significantly. In order to increase the efficiency of the test and to lower the costs, test case design has to be systematized and automated. However, the embedded systems are often subjected to change to meet customers need and functionalities, so that it is desirable to automate testing to improve the efficiency of process. Since there are huge number of test cases to execute in embedded systems, automation is required to maximize coverage and reliability. Test-automation is a must to reduce expensive human efforts. Therefore, a lot of testing tools are on the market specialised in test automation. The main emphasis of the tools is the automation of test execution, monitoring, and test documentation. Infrequently, support for test case design is offered.

System testing of electronic control units is usually black-box-testing. Test cases are manually derived from the specification. Various works propose the use of evolutionary testing for the automation of test case design. Most common is the automation of structural test case design, but even for the automation of structural testing evolutionary testing [13] is not widely used in industrial practice due to the missing tool support. In practice, even more important is functional testing validating the system under test against its specification. Functional testing using search-based techniques is not widely common, because efforts for the development of the fitness functions and for setting up appropriate test environments is high.

In this work we develop a testing solution which supports search-based functional testing for hardware-in-the-loop (HiL) testing in a uniform way. This paper present an approach in which evolutionary functional testing is performed using an actual electronic control unit for test case evaluation. A test environment designed to be used for large-scale industrial systems is introduced. The testing framework developed has been evaluated for testing the functioning of a serial production of automation-system (AS) electronic control unit in an industrial setting.

II. RELATED WORK

Automation approach is to apply evolutionary testing technique, thus transforming the testing problem into an optimization problem to be solved using evolutionary algorithm (EA). Research over recent years has already demonstrated the successful application of EA for the functional testing. However, the need for executing the system under test using continuous test data has not sufficiently been addressed therein. The potential of performing search-based functional testing on various test platforms. They demonstrated the practicality of evolutionary functional testing without extensively carrying out a real case study which is the aim of this paper.

A. Evolutionary Functional Testing

Functional testing aims at validating the functional behaviour of a software system. The tester is supposed to find test data that violates the functional requirements of the system under test. This task is very cost-intensive, time-consuming and error-prone when done manually, thus the automation of this process is highly aspired.

Revised Manuscript Received on 30 June 2012.
* Correspondence Author
Miss. Ashwini M. Motghare, Department of Embedded System & Computing, G. H. Raisoni College of Engineering Autonomous Institute. Nagpur (M.H), India.
Prof. Swapnili P. Karmore, Department of Embedded System & Computing, G. H. Raisoni College of Engineering Autonomous Institute. Nagpur (M.H), India.

Evolutionary functional testing allows for the automation of functional testing by transforming the task of generating relevant test data into an optimization problem and tries to solve it using a meta-heuristic search technique [8]. Search techniques such as Evolutionary Algorithm (EA) are used to find approximations to the optimal solution in multidimensional search spaces. These general purpose search techniques make very few assumptions about the underlying problem they are attempting to solve. As a consequence, they are useful during the automated generation of effective test data.

A working test environment provided, three domain specific components have to be supplied for each test goal in orders to perform an evolutionary functional test. First, a suitable fitness function has to be derived from the functional specification of the system under test. The fitness function evaluates every test run with respect to its criticality and assigns a fitness value to it. This fitness value is used to compare multiple test data sets with each other and to optimize the data sets towards a failure of the system behaviour. The second problem that has to be tackled is defining the search space.

When testing complex embedded systems the execution time for a single test run can take up a considerable amount of time. Effort has to be put into limiting the search space to the important data ranges to make the search feasible. Finally, a test driver has to be implemented. The task of the test driver is to map the generated test data to the input ports of the system under test and to execute it.

III. TEST ENVIRONMENT

The test environment or Evolutionary Testing Framework
(ETF) has been developed in the context of EvoTest [13]. In the first section of this chapter describes the basic functionality of the Evolutionary Test Framework (ETF), whereas second section describes the experimental testing environment necessary for facilitating HiL testing.

A. Evolutionary Testing Framework (ETF)

Electronic control units are used in nearly all industrial areas to control complex systems like airplanes, cars, trains, engines etc. Usually, testing of such systems contains unit testing, integration testing and system testing. Common test platforms for the testing are model-in-the-loop testing (testing the model or parts of the model in a simulation environment), software-in-the-loop testing (testing the resulting software in a simulation environment) and hardware-in-the-loop testing (testing the software integrated on the electronic control unit in a real-time simulation environment).

The Evolutionary Testing Framework has been used to perform the case study. It provides general components and interfaces to facilitate the automatic generation, execution, monitoring and evaluation of test scenarios. It is designed to be used with large and complex systems which are common in industrial settings. The underlying evolutionary computation techniques are hidden through a user interface. This is an important practical component because it allows practicing software engineers to use the ETF without requiring any knowledge of evolutionary computation.

Figure 1 shows the general workflow of the ETF. The framework has to be capable of generating realistic input signals due to the dependence on realistic input signals of the system under test. This task is done by the signal generator, which in turn expects a signal representation from the tester. This signal representation describes the nature of each input signal to be generated. Based on the signal representation the signal generator is able to create and provide the optimization engine with a description of the problem or the search space, respectively. According to this, an optimization engine is generated and compiled in order to optimize the problem. The optimization engine will create and initialize the first population of individuals which are sent to the signal generator. The signal generator uses the provided set of individuals in combination with the specified signal representation to generate a number of signal sets. These signal sets are passed to the test driver. The test driver then executes the system under test with the provided signal sets and records the resulting output signals.

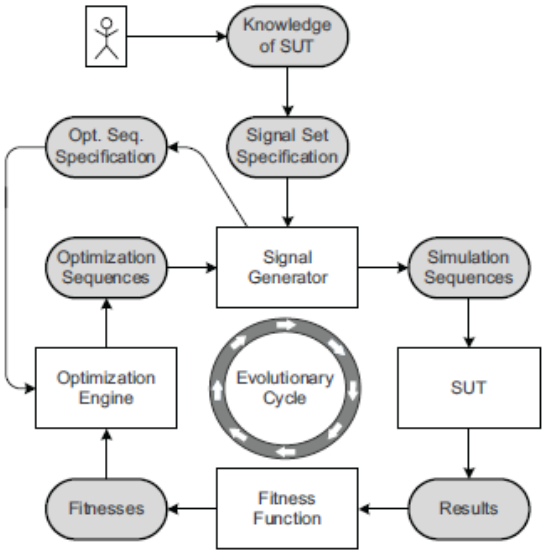

Figure1. System Architecture of the Test Environment

The results which represent the system behaviour for the respective input scenarios are then evaluated using the Fitness Function. The monitored set of observation signals is evaluated for each enclosed time step. The *Fitness Evaluator* calculates one fitness value for each time step using distance metrics, based on the observations measured during experimentation.

The fitness values for each input signal set are passed to the Optimization Engine, which creates a new population of individuals based on this assessment. This closes the evolutionary cycle and the optimization continues until the demanded solution has been found or another stopping criterion applied.

B. Hardware-in-the-Loop Test Platform

Hardware-in-the-loop testing systems are necessary to test electronic control units in a simulation environment in real-time. In order to test electronic control units their interfaces are connected to the hardware-in-the-loop test system providing the corresponding input signals and reading the output signals. Usually, a simulation environment is necessary to run the tests and to simulate the real application environment of the electronic control unit.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

As an individual module, with flexible signal conditioning and the use of open industry standards, modularHiL could be used for testing a single embedded control unit. Using an optical networking technology it is possible to link several modularHiL systems together to form a powerful test environment for integration testing. During integration testing the interplay of several electronic control units is tested. Because it uses the same modules for component and integration test systems, modularHiL offers customers cost benefits and high flexibility.

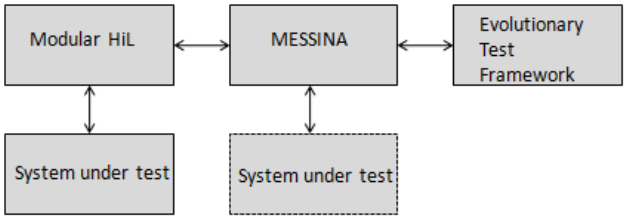

Figure2. Component of the test Automation Framework

MESSINA is a testing infrastructure allowing the implementation of hardware and software independent test sequences in different notations such as UML, Java, or TPT. Using two abstraction layers it offers execution on different platforms. The first layer is the signal pool containing all system signals provided by the connected devices or software devices. The signal pool allows easy read- and write-access to all the signals used by the system under test or the simulation environment. The second layer is formed by abstract in-the-loop systems, which can be MiL, SiL and HiL systems.

The model has to be tested to ensure a good specification quality. Software testing is performed to verify that the software implementation fulfils the specification. If code generation is performed, software testing ensures the correct transformation performed by the code generator from model to software. Integration testing of hardware and software is always performed to verify if the integrated system works correctly.

For MiL and SiL testing MESSINA supports software devices like MATLAB/Simulink models, and AUTOSAR software components. For HiL testing MESSINA is directly connected with modularHiL. It is possible to download tests implemented with MESSINA to the modularHiL where the tests are executed in real-time. As a result tests implemented in MESSINA can be used seamlessly in the model test (MiL), software test (SiL) and hardware test (HiL). Therefore, MESSINA could be used for thorough model-based electronic control unit (ECU) testing from specification to HiL testing.

C. Test Environment Architecture

Embedded system testing environment (ESTE) is a kind of computer system that orients to embedded system testing. Using ESTE, tester can organize the inputs of SUT, complete configuration, drive test process and collect the output of SUT in order to realize the real-time testing for the embedded systems.

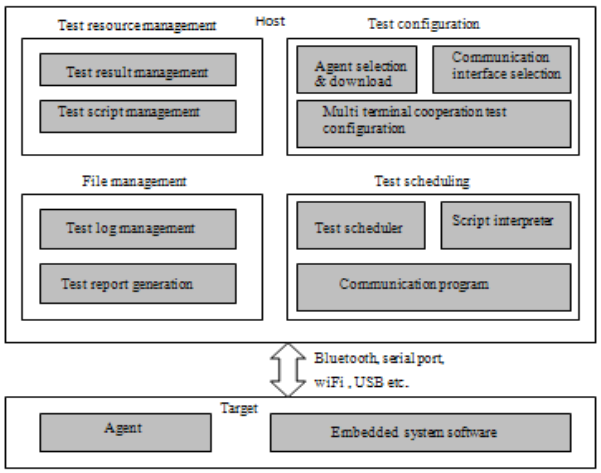

Figure3. Test Environment Architecture

The functions of testing environment host are

1. Test Scheduling - Completing the monitoring and scheduling during test process, and realizing the automatic testing with nobody by the use of script-driven method.
2. Test Resource Management – Managing test resources during test process, depositing and getting test results and recording the test script.
3. Testing Environment Configuration – allowing the tester to configure the testing environment such as selecting and downloading the agent, selecting the interfaces between host and the target machine, and configure the multi terminal cooperating testing.
4. File Management – Recording the testing log information and forming test report.

IV. EVALUATION

Growing Technology and their development increasingly influence Automation in industry. And, it plays an important role in the global economy and in our daily lives. Almost all the process monitoring systems installed as a part of plant or production process are basically Digital Control Systems DCS connected by digital networks. The purpose of automation has shifted from increasing productivity and reducing costs, to broader issues, such as increasing safety, quality and flexibility in the manufacturing process. With the help of DCS and wireless network, it is possible to make both startup activity and operational routines of a complicated process much easier and more efficient. DCS also offer process modeling and simulation.

We demonstrate the use of evolutionary testing for functional testing in an industrial setting by applying the developed solution to the testing of Process Flow Automation System through Simulating Environment.

Developed system consists of four servo motors whose functioning is to monitor and test serial production of process flow automation system. System also consists of input channel which is connected to proximity sensors for giving input signals to the system. Task of proximity sensors is to provide input signals in the form of 0's and 1's (low and high) to the input channels.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Depending on the input signals given by input channels corresponding servo motors work on. By setting the sequence of servo motors developer can set numbers of functional task. Since there are four input channels so user can get 16 different combinations.

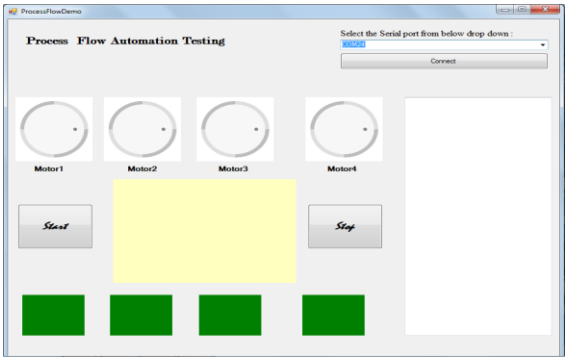

Figure4. Process Flow Automation System in Simulating Environment

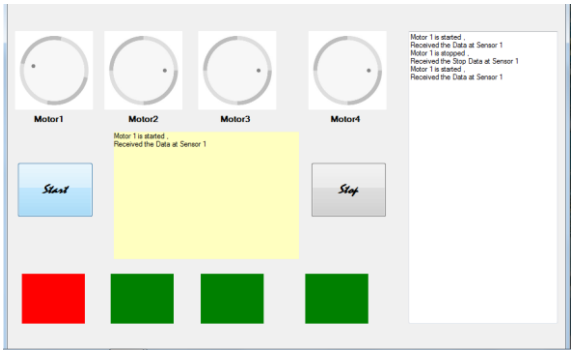

Figure5. Result of Process Flow Automation System in Simulating Environment

Fig. 4 describes process flow automation system in simulating environment. To test sequence of process flow automation in simulation Testing Environment, firstly it is connected to simulation testing environment through serial port as per the given APIs of the selected device. By Selecting corresponding port tester can make connection to device. After clicking on start button of testing environment user can test sequence of process flow automation in simulation Testing environment. And result of working corresponding servo motors will be display on simulating test environment as shown in Fig.

V. CONCLUSION

In this work test automation framework has been developed that allows full automation of functional testing on different testing platforms (MiL, SiL, and HiL). The provided solution supplements systematic testing by reducing the risk of non-testing situations unforeseen by the testers. The test automation framework supports the application of evolutionary testing in very common industrial settings: testing models and software in simulation environments as well as the examination of electronic control units in a hardware in- the-loop test environment driven by a software frontend for the definition and implementation of tests. To evaluate the test automation framework testing a process flow system formed the first case study. Test cases were generated using the evolutionary testing framework and executed in SiL and HiL test environments.

REFERENCES

1. Byeongdo Kang, Young-Jik Kwon, Roger Y. Lee, “A Design and Test Technique for Embedded Software”, IEEE, 2005.
2. Pei Tian, Kai Wang, Kai Qiang, " Construction of Distributed Embedded Software Testing Environment", IEEE, 2009.
3. Dae-Hyun Kum, Joonwoo Son, Seon-bang Lee, “Automated Testing for Automotive Embedded Systems”, SICE-ICASE, 2006.
4. Yongfeng YIN, Bin LIU, Guoliang ZHANG, “On Framework Oriented Embedded Software Testing Development Environment”, IEEE,2009.
5. Kandl, S.Kirner, R. Puschner, "Development of Framework Automated Systematic Testing of Safty-Critical Embedded Systems”, IEEE, Intelligent Solutions in Embedded Systems, 2006.
6. D. L. kaleita and N. Hartmann, "Test Development Challenges for Evolving Automotive Electronic Technologies", SAE, 2004-21-0015, 2004.
7. Bringmann, E.Kramer, “Model-Based Testing Automotive Systems”, IEEE, Software Testing, Verification, and Validation, 2008.
8. Lindlar, F. Windisch, “A Search-Based Approach to Functional Hardware-in-the-Loop Testing”, IEEE, 2010.
9. Siegl, S.Caliebe, “Improving Model-Based Verification of Embedded System by Analyzing Component Dependences”, IEEE, Industrial Embedded Systems (SIES), 2011.
10. N. H. Lee and S. D. Cha, "Generation Test Sequences from a Set of MSCs", The International Journal of Computer and Telecommunications Networking, Volume 42, Issue 3, Page 405 - 417, 2003.
11. Y. G. Kim, H. S. Hong, D. H. Bae and S.D.Cha,"Test Cases Generation from UML State Diagrams", IEE proceedings, online no. 199990602, 1999.
12. R. L. Probert, H. Ural and A.W.Williams, "Rapid Generation of Functional Tests using MSCs, SDL and TTCN", Computer Communications, Volume 24, Issues 3-4, Page 374-393, 2001.
13. Wegener, J.; Baresel, A.; Sthamer, H.; “Evolutionary test environment for automatic structural testing”, Information and Software Technology, 2001.

AUTHOR PROFILE

Ashwini Motghare is student of M. E. in Embedded System & Computing at G. H. Raisoni College of Engineering Autonomous Institution, Nagpur Maharashtra.

Prof. Swapnili P. Karmore is Professor of Embedded System & Computing at G. H. Raisoni College of Engineering Autonomous Institution, Nagpur Maharashtra.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)