Long Term Research Agenda

Abstract

This report is part of the CIRENE project that aims the definition future work including FP7 project proposal, research topics for MSc and PhD students, and the possibilities to integrate CIRENE project results in the education.

This work were done in the Cross-border ICT Research Network (CIRENE) project (project number is HUSRB/1002/214/044) supported by the **Hungary-Serbia IPA Cross-border Co-operation Programme**, co-financed by the European Union.

Table of contents

1 Introduction

In the CIRENE project we have defined a general methodology for embedded systems testing, and implemented it on a specific system resulting in a working prototype. In this document we summarize how these achievements of the CIRENE project can be used for further research, development and/or teaching activities.

In Section 2 an EU FP7 project proposal idea is introduced. It aims the research of test automation of digital multimedia devices (special embedded systems). It is connected to the embedded testing methodology in the steps test case generation and execution.

Section 3 describes some more concrete topics that require much less effort than the previous one. These topics are more directly based on the achievements of CIRENE, and small enough to be research topics for MSc or PhD students.

Last, but not least in Section 4 we describe how the achievements could be included in the teaching activities of the universities.

2 EU FP7 project proposal

2.1 Proposal: Reliability enhancement of multimedia content distribution by automated test approaches

2.1.1 Introduction

Mass production of new multimedia devices for rich home entertainment, specifically HDTV (High-Definition Television), enabled the broadcasting of higher quality video material (cable, terrestrial or satellite broadcast) and additionally, broadcasting over IP networks. In order to provide a digital audio and video content to the end-users, initially the STB (Set-Top Box) devices were used. At that time, the basic functionality of STB was demodulation and decoding of DVB (Digital Video Broadcast) to audio-visual content suitable for recording or displaying on an analogue TV/monitor.

Over the years, STB devices have progressed in terms of their functionality related to new interactive and non-interactive services and applications. Accordingly, the complexity of the architecture and functionalities of STBs has increased, while functional testing became much more difficult. At the same time, the demand for fast and efficient functional verification increased, and it became more important, given that the testing is done several times at different stages of the manufacturing process, before the product reaches the market. In order to reduce the time to market (TTM), and considering that manual testing requires a big effort from a man, may take a long time, and is subject to errors, the approach of automatic functional verification came as a natural choice for efficient development of high quality multimedia devices. Hence, an automated testing procedure is crucial for manufacturers of STB devices. However, designing the system for testing and the implementation of automated testing, is in whole a complex and difficult task due to challenges related to automatic test case generation for a wide range of functionalities , automatic test execution on the STB under

2.1.2 Statistical and search based testing based on operational profile and code coverage

One approach for automated generation of test cases is statistical testing by using operational profiles. Considering IP as the most popular broadcast method for STBs and DTVs at the moment, devices from this group present the latest challenge for testing. Thus, the definition of the operational profile has to be suited for IP STBs or DTVs. Based on predefined operational profile it is possible to automatically generate test cases using available tools like MaTeLo. Automated test case generation, test execution, and test results collection completes the path of automated testing. Test results are used in further analysis and final calculation of reliability and mean time to failure measures.

Operational profile-based test generation can be applied to various test types including generation of audio and video streams, IP network traffic load testing and similar. Such specific tests could be economically performed by using a simulator on the network traffic level, which could act as a proxy server which interferes between the clients (digital receivers) and the server.

In addition, the required tests can also be prepared based on code coverage information. However, successful profile-based test generation could still be part of the implementation not exercised by

testing. Hence, code coverage can be used to predict the reliability of the system and to drive additional test generation strategies. For instance, search based methods can help to generate the specific inputs to achieve required execution paths through the software.

Additional use case for such testing environment can also include states for playing streams to the device under test, running video quality assessment on the output of the device and producing test results according to extracted measures. The video quality assessment can be based on calculation of artifacts that are caused due to video coding/decoding process such as blocking, blurring, ringing and can be extended to simulate transmission errors such as packet loss. It is possible not only to determine the system reliability and mean time to failure, but to obtain information of coverage requirements and failure intensity as well.

The proposed technology will contribute to the state-of-the art in testing multimedia broadcast systems and will provide higher reliability to these systems and more cost effective development and production life cycles of IPTV STBs.

2.1.3 Status

We are looking for research institutions and industrial partners joining the consortium, who have competence in the fields of digital multimedia broadcasting, embedded systems development, software testing and quality assurance. Prospective users of the technology in pilot studies are especially welcome.

2.2 Submission possibilities

We investigated fp7-ict-2013-11 and fp7-ict-2013-c calls but they were not appropriate. Unfortunately currently there are no appropriate FP7 calls. We expect more calls in the new budget period of the EU (2014-2020).

3 Research Topics

There are many research topics that can build on the results of the CIRENE project. Some of these are small enough to be the base of the thesis of an MSc or PhD student. So, we have defined some topics and will publish them in both the University of Novi Sad and the University of Szeged. The goal of this parallel publication is the stimulation of the cooperation of the students from the two universities. If the same topic is selected by two students, one from Novi Sad and one from Szeged, then the students will be asked to work together in a joint work on the same topic. This way the topics strengthen the collaboration and partnership between the universities.

3.1 MSc topics

3.1.1 Traceability information

Determine traceability links between program GUI elements, program functionalities and source code using coverage information.

We can collect coverage information from the source code using instrumentation. By linking coverage to test cases and linking test cases to program functionalities, we can ascertain the link between source code and program functionalities. If a test case includes the activation of some GUI elements, we can use this information in a similar way to detect relationship between GUI elements and source code.

The difficulty of these tasks is that the links are not necessarily transitive and straightforward. For example, the main method itself will probably not link to any specific functionalities or requirements even if it is executed by all test cases. Finding such links is thus a complex problem that probably does not have an unambiguous solution: we should work with probabilities and thresholds. The goal is to find methods and models that can help defining the above mentioned traceability links based on the available coverage data.

3.1.2 Detailed call tree building for Android applications

Log method entries/exits and build a call tree for android applications. Handle threads, exceptions. Dump special information from some selected methods (e.g. coordinates from action handling methods).

Dynamic call trees can be used for comprehending the programs actual behavior and in this way they can play a role in test case generation. Our current solution logs method entries only, and it is not prepared to handle parallel executions of the same method, thus it can be used to build a call tree just in special cases.

The goal of this work is to develop and build an instrumentation technique that builds up a detailed call tree for the applications. The solutions should probably include the logging of method entries, normal and exceptional method exists, thread information. Logs will contain various common data like timestamps of the events. In order to give more support for black-box testing, the solution should be able to log some attributes/parameters of special kind of methods (e.g. action handling method position parameters).

3.1.3 Instruction level coverage measurement of Android applications

Create a framework for instruction level coverage measurement of Android applications.

Our current solution works on method level. It is fast (has only a minimal impact on execution time of the application) but produces coarse grained information. It is necessary but not enough to call all methods during the tests: executing only one of the multiple execution paths of a method results in that the method is treated as covered, but many instructions of it are not exercised.

The goal is to develop an instrumentation and coverage measurement method that is able to uniquely identify all basic blocks or individual instructions of the program, is able to log the execution of them, and is effective in the means that it does not have a heavy influence on the execution time.

3.1.4 Branch/decision level coverage measurement of Android applications

Create a framework for branch level coverage measurement of Android applications.

Our current solution works on method level. It is fast (has only a minimal impact on execution time of the application) but produces coarse grained information. It is necessary but not enough to call all methods during the tests: executing only one of the multiple execution paths of a method results in that the method is treated as covered. Instruction-level coverage is still a necessary but not sufficient condition in most of the cases: sometimes it is necessary to test what happens when the given instructions are definitely not executed.

The goal is to develop an instrumentation and coverage measurement method that is able to uniquely identify all decisions and branches of the program, is able to log the results of the decisions and the execution of the branches, and is effective in the means that it does not have a heavy influence on the execution time. As an option, the solution should consider the measurement of other branch-level coverage values (condition, multiple condition, etc.).

3.1.5 Offline coverage measurement for Android applications

Find solutions for offline coverage measurement on Android.

Our current solution assumes live network connection. However, in some cases the storage of such data for longer times would be more preferable (or would be the only way). Thus, the data collection mechanism should be modified or replaced in order to effectively store the generated data on the device.

The goal is to develop and build a technique or a methodology that can be used for offline coverage measurement by storing large amount of coverage data effectively on the device being measured.

3.1.6 Automatic test case generation based on given usage model

The goal of this topic is to implement different test strategies for automatic generation of adequate number of test cases, based on defined abstract usage model of the system to be tested. The number of test cases depends on concrete test strategy (Most probable path, Given reliability, Maximal coverage, etc.). Test strategies have to enable the generation of test cases which will cover all parts of the system, and to determine the number of test cases that could guarantee satisfactory reliability

of the system. For determining the number of test cases, and also for determining minimal/maximal reliability of the system that is needed to be checked, different mathematical or statistical models are used (Weilbul, Posaon, etc.).

3.1.7 Black-box testing of touch screen devices

The goal of this topic is to develop appropriate environment for graphical creation of test cases intended for testing of touch-screen devices. Specifically, it is necessary to create an environment for drawing actions that user want to emulate (dunning application, listing, scrolling, multi-touch, etc.). Besides, the method for creating these test cases should include a way to run test cases during their creation. That way the user can see (preview window with image from touch-screen device display should be included) current picture, and position his actions accordingly. Test cases are validated at the same time. The last thing is to create test scripts based on given graphical model.

3.1.8 Test execution controllers for Android-based portable devices for testing television equipment on the field

In this topic, environment for testing multimedia devices (such as Digital TV receivers, STBs, etc.) but adjusted to touch-screen devices should be developed. It should include main application intended for selecting and running test cases. The application should include a module for interfacing different services declared in testing procedure. The set of services for controlling components of the testing system should be implemented, also for Android based devices.

3.1.9 Operational profiles for multimedia streams, and Internet traffic simulations aiding black-box testing

The goal of this research topic is to develop a family of realistic operational profiles that will aid holistic testing of multimedia devices, such as Digital TV receivers, STBs, etc. The state of the art operational profiles are used for testing individual functional subsystems of these devices, and mostly graphical user interfaces. Other parts of the target systems are randomly tested. In this research we plan to test target devices completely by designing operational profiles that will be used to generate all possible inputs, including internet traffic and multimedia streams.

3.2 PhD topics

3.2.1 Analysis of profiling information for Android applications

Measure the execution time, count and frequency of different program elements (methods, basic blocks, cycles, etc.). Compare this information with other (static) metrics (NII, NOI, Complexity, …). Find and analyze connections, correlations.

Profiling information (time spent in code segments, execution count of loops, etc.) can be used in test case generation, test selection and prioritization: tests covering more frequently used or slower execution paths should have higher priority. Gathering profiling information requires logging of executed program constructs, execution times, etc. during test execution. In some cases it would be more preferable if some less time consuming measurements (e.g. static metrics) could predict profiling results. It is a research task to check possible relations and correlations between program element profile results and static metrics. If there are some relations, even static metrics could be

used in test case selection and prioritization, which reduces the “chicken and egg” problem of using profiling information for the tests that will generate this profiling information.

The goal is to develop and build a technique for Android platform that can be used for measuring execution time, count, frequency and other relevant information of different program elements (instructions, methods, basic-blocs, loops, branches, decisions, etc.). Compare the gathered information with various static metrics (NII, NOI, Complexity, etc.). Create a framework for comparison and analysis of connections and correlations in this data. Check how the findings could be used in test selection and test case prioritization.

4 Integrate results in education

In this section we present how the achieved results can be used in education, how these can be integrated in the training programs of the universities.

4.1 Testing Methods

The topics of this course of the University of Szeged cover different advanced techniques that can be used to improve software testing. These topics include the general test process, coverage measurement, test prioritization and selection. The testing methodology can be used to demonstrate a testing process. The prototype tools could be used to demonstrate the full test cycle with selection, prioritization, and execution on specific systems. Students could have the choice to use this toolset for their project work on this course.

4.2 Android development

In this course students are gaining knowledge on Android specific development. They could use the prototype toolchain to aid their development work by acquiring coverage or (after some modification of the toolchain parts) even profiling information. Requiring the students to utilize the measurement framework in their project works could provide some statistical data for future investigation.

4.3 Embedded systems

This is similar to the Android development course except that it concentrates more on hardware specific programming. Here the methodology could be presented to the students, and besides the usual project works, students are offered to implement different part of the framework for that specific system.