PhD in Information Technology and Electrical Engineering

Università degli Studi di Napoli Federico II

PhD Student: Vincenzo Riccio

XXXI Cycle

Training and Research Activities Report – First Year

Tutor: Anna Rita Fasolino

Information

I am Vincenzo Riccio, I obtained a Ms.Sc. Degree (Laurea Magistrale) cum laude in Computer Engineering (Ingegneria Informatica) at the Università degli Studi di Napoli Federico II in April 2015. I am a First Year PhD Student attending the XXXI Cycle of the Information Technology and Electrical Engineering (ITEE) PhD program of Università degli Studi di Napoli Federico II. My fellowship is financed by a PhD student grant. I am carrying out my research activity within the RevERSE research group of Software Engineering under the tutorship of Prof. Anna Rita Fasolino and collaborating with Domenico Amalfitano, Porfirio Tramontana, Nicola Amatucci, and Vincenzo De Simone.

Study and Training activities

In the first year of PhD program (1.11.2015-31.10.2016), I attended the PhD courses and carried out the training activities reported in tables below; these activities include Modules, Seminars, and an IEEE Doctoral Summer School at University of Salerno.

Modules

Name	Type	Provided by	Credits
Specification and Verification of Multi-Agent Systems	Occasionally Provided	Prof. Wojtek Jamroga (Polish Academy of Sciences)	3
Enterpreneurial Analysis of Engineering Research Projects	Ad hoc module	Prof. Luca Iandoli	3
Communicating and disseminating your research work	Ad hoc module	Prof. Mo Mansouri	3
Modelli e metodi di Ottimizzazione*	Ad hoc module	Prof. Antonio Sforza	4
Software di Ottimizzazione*	Ad hoc module	Prof. Antonio Sforza; Ing. Claudio Sterle	4
Scientific Writing	Ad hoc module	Prof. Paolo Russo	5
12th International Summer School on Software Engineering (IEEE-ISSSE 2016)	Doctoral School	Prof. Andrea De Lucia	3
(*) Modules for which I attended the course but I have not taken the exam yet			

Seminars

Name	Type	Speaker	Credits
Memory Technologies for Android based Systems	Seminar	Luca Porzio	0,4
Test and Diagnosis of Integrated Circuits	Seminar	Alberto Bosio	1,4
Networks on Chip: Introduction and Advanced Topics	Seminar	J. Flich	0,8
Hardware Security and Trust	Seminar	Giorgio Di Natale	2,8
Model based and Pattern based GUI Testing	Seminar	Ana Paiva	1*
Verifica e Validazione di sistemi Safety Critical	Seminar	A. Di Nocera; G. D'Avino	0,4
Radar Adaptivity: Antenna-based Signal Processing Technique	Seminar	Alfonso Farina	0,4
Perception-based surround sound recording and reproduction	Seminar	Enzo De Sena	0,2
Adversarial Testing of Protocol Implementations	Seminar	Cristina Nita Rotaru	0,4
Programmable Network Conjugations	Seminar	Roberto Bifulco	0,4
Speech Technologies at Trinity College-Dublin	Seminar	Loredana Cerrato	0,2
Challenging real-time measurement systems for immersive life-size augmented environment	Seminar	Giovanni Caturano	0,5
Internet: la dimensione immateriale dell'esistenza	Seminar	Stefano Quintarelli	0,4
Methodologies for Embedded Software Validation	Seminar	Diego Tornese	0,2
Microsoft Azure	Seminar	Sandro D'Aviera	0,2
La protezione brevettuale: Opportunità, Procedure, Casi di Studio	Seminar	B. Bjola; M. Maremonti	0,5
(*) 0,2 extra credits for supplementary activities			

Research Activity

In my PhD I am deepen my knowledge in the subject of Software Engineering. My main research topic is Software Testing, a practice that allows to evaluate and improve the software quality. In highly competitive sectors where software is a key component of the product, there is a constant pressure to improve the software quality and to quantitative evidence of the improvement. I focused on software testing in automotive and mobile domains.

Coverage Testing of Software auto-generated from Models

This research activity concerns the novel approaches in software development that focus on implementing models of software systems that are automatically transformed into target code. They are emerging as a relevant research topic in both industrial and scientific communities as they shift the focus of software development from writing code to modeling the behavior of the system [1].

Testing is a common practice to assess and improve software quality. In Model based approaches, models can be tested before the target code generation; this allows detecting earlier software faults and reducing the costs of software development processes. However, the quality of code obtained from models through code generation tools is not assessed by the quality measured on the same models by the testing process [2]. Therefore, both models and target code are tested in critical software development processes, e.g. automotive safety software.

Coverage analysis allows measuring the goodness of a software testing process by evaluating the software covered by the execution of the test cases. The main objective of the proposed research activity is to evaluate the differences that may exist between the model coverage guaranteed by the test cases and the code coverage reached when the same test cases are executed on the auto-generated code. Moreover, we identify the main factors that may influence these differences.

I gathered my first experiences in this research topic during a stage at FIAT Chrysler Automobiles s.p.a. for my Master Thesis *"Un tool per l'analisi di copertura di codice C autogenerato da modelli MATLAB/Simulink per sistemi embedded in ambito automotive"*. Then , I enhanced my knowledge thanks to my involvement in the CeRICT s.c.r.l. activity *"Analisi e implementazione di tecniche di coverage testing a supporto della fase di V&V del ciclo di vita del software embedded in conformità con la norma ISO 26262-6"* in the project PON03PE_00159_7 named *"APPS4SAFETY: Metodologie e tecnologie innovative per un approccio integrato alla sicurezza del veicolo"*. The main goal of these works was to develop techniques and tools for supporting the Verification&Validation process of embedded software in the automotive domain. I focused my efforts towards the automation of the tasks required to evaluate the coverage of the embedded code auto-generated from models.

This background allowed me to support the preparation of the paper **"Comparing Model Coverage and Code Coverage in Model Driven Testing: An Exploratory Study"** [3], presented by D. Amalfitano at 6th International Workshop on Testing Techniques for Event BasED Software (TESTBEDS 2015) co-located with 10th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015). Model Driven Testing (MDT) [4] is a software testing methodology where test cases for the system are automatically obtained starting from test models to maximize specific model coverage criteria. In MDT, the models are described in UML, the OMG standard modeling language. Eventually, test cases are executed to verify the system code that is automatically generated from models. The main goal of this paper is to evaluate the differences that may exist between the model coverage guaranteed by

the test cases and the code coverage reached when they are executed on the auto-generated code. Moreover, we wanted to identify the main factors that may influence these differences.

Thus, we conducted an exploratory study implementing a model driven approach using UML StateMachine (FSM) models. We considered the finite state machine (FSM) model because it is appropriate to specify the behavior of event-driven systems, including Graphical User Interfaces (GUIs) and control systems. The results of our study showed us that there are differences between model coverage and code coverage. We were able to understand that the main factors influencing differences in model and code coverage of test cases automatically produced in Model Driven approaches are (1) the tool implementing the transformation rules from the model towards system code, (2) the test adequacy criteria at model level exploited for the test code generation, and (3) the style adopted by the modeler to design the system behavior.

Automated Testing Techniques for Mobile Applications

The major research activity of my first year concerns Mobile Software Testing Automation, one of the main topics addressed by RevERSE research group. Software testing is a well-known approach for assuring the quality of mobile applications. Mobile applications present new challenges in contrast to other types of applications due to the peculiarities of the mobile world, limited resources, new development concepts, context awareness, and the diversity of devices and their characteristics [5].

Accordingly, it is necessary to study how to adapt the existing testing approaches to mobile apps or even to define new specific approaches. A relevant part of existing techniques and tools emphasize testing the functionality of a system through its Graphical User Interface (GUI) [6]. These techniques are event-based but often neglect mobile-specific events, such as putting an application in background and resuming it, receiving a call, rotating the device [7].

Most approaches aim to maximize the code coverage or to find crashes in the apps under test. Few of them face the problem of detecting unexpected states of the GUI, referred to as GUI failures [8].

GUI Testing emerged as a common research ground with Prof. Ana C. R. Paiva from the Informatics Engineering Department of the Faculty of Engineering of University of Porto (FEUP) who held the seminars *"Model based and Pattern based GUI Testing"* in November 2016, organized by Prof. Anna Rita Fasolino and Prof. Porfirio Tramontana as part of the ITEE PhD program. This meeting was an opportunity for the RevERSE research group to start working together with the Prof. Paiva's research group in order to address the problem of GUI failures due to mobile-specific events.

Our efforts aimed at finding GUI failures, classifying them, and understanding their causes. We focused on GUI failures due to orientation change events since affect several apps and mobile OSs. When the user rotates the device, a mobile application should respond accordingly, adapting itself to the new layout, avoiding memory leaks, and preserving its state and any significant stateful transaction that was pending. Android provides recommendations and guidelines that show the programmers how to manage this event [9,10]; but many mobile apps crash or show GUI inconsistencies that can be attributed to orientation change mishandling. It can be observed that applying a single orientation change may not be sufficient to detect GUI failures as some minor differences in GUI content or views are indeed acceptable between landscape and portrait orientations [9]. Instead, after a second consecutive orientation change, the GUI content and layout should be the same as before the first orientation

change, otherwise we have a GUI failure. Therefore, we exploited the double orientation change event sequence for testing Android apps, with the aim of finding GUI failures.

Starting from a preliminary investigation involving apps from different mobile platforms, we defined a high level GUI failures model in order to classify them. This model abstracts three main classes of failure that may be triggered by a double orientation change. These failures include unexpected disappearing of graphical/textual objects from the GUI, unforeseen appearing of objects or unpredicted state changes of GUI objects.

We also proposed a black-box testing technique for detecting them. This technique was exploited in an exploratory study that involved 50 open source Android applications available on the official Android app store. The apps were tested by two different GUI-based testing strategies that exercised the double orientation change event sequence. We found that 82% of the analyzed apps presented GUI failures due to the orientation change. We classified the detected failures according to the proposed model and evaluated the distribution of different GUI failure types and the GUI objects more frequently involved in these failures. Moreover we analyzed the source code of the apps, in order to locate the faults causing the failures.

The results of the study pointed out that this problem is widespread in the context of Android mobile apps and worth to be investigated. Moreover, we recognized that some types of failures are more common than others and some GUI object types are more frequently involved. Our study also identified common faults in the apps source code that may cause the addressed GUI failures and should be avoided by developers. This study has been described in collaboration with D. Amalfitano, I. Coimbra Morgado, A. R. Fasolino, and A. Paiva in the paper **"Exploring GUI Failures in Mobile Applications due to Device Orientation Change"**, submitted on September 2016 to the Journal of Software: Evolution and Process, edited by Gerardo Canfora, Darren Dalcher and David Raffo, published by Wiley, impact factor 0,729. This activity is open to future works as extending the validity of our result by carrying out an experiment involving a greater number of apps, other mobile operating systems, and other mobile-specific events which may cause GUI failures, such as Receiving a Call, Sending an application in Background, Pressing the Back button of the device. Moreover, we could focus on developing fault localization techniques focused on source code bugs that may cause these failures.

Moreover, another aspect of mobile testing that we considered is **Robustness Testing**. The IEEE/ISO/IEC 24765:2010 Standard defines the robustness as the degree to which a system operates correctly in presence of exceptional inputs and stressful conditions [11]. This testing approach aims to find the faults that undermine the robustness of the system. We adopted the Negative Testing approach which aims at showing that a software does not work [12] by identifying faults that results in crashes. We are focusing in implementing a technique that tests the robustness of Android apps at system level. We are currently working in implementing this technique in a tool to present at the Tool Demo Track of 10th IEEE International Conference on Software Testing, Verification and Validation (ICST 2017). This tool will be exploited in the study **"aRTETECA - Robustness TEsting TEChnique for Android: an Exploratory Study".**

Products

Conference Papers

[P1] D. Amalfitano, V. De Simone, A. R. Fasolino and V. Riccio, **"Comparing Model Coverage and Code Coverage in Model Driven Testing: An Exploratory Study"**, 2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), Lincoln, NE, 2015, pp. 70-73. doi: 10.1109/ASEW.2015.1830.

Submitted Papers

[P2] D. Amalfitano, I. Coimbra Morgado, A. R. Fasolino, A. Paiva and V. Riccio, **"Exploring GUI Failures in Mobile Applications due to Device Orientation Change"**, submitted on 26 September 2016 to the journal Journal of Software: Evolution and Process, Wiley.

Papers in preparation

[P3] D. Amalfitano, V. De Simone, A. R. Fasolino and V. Riccio, **"aRTETECA - Robustness TEsting TEChnique for Android: an Exploratory Study"**.

[P4] D. Amalfitano, V. De Simone, A. R. Fasolino and V. Riccio, **"aRTETECA: Tool Demonstration"** (ICST 2017 Tool Demo Track).

Conferences and Seminars

I have not made any presentation at conferences or seminars in my first PhD year.

Activity abroad

I do not have carried out any activity abroad in my first PhD year

Credits Summary

Student: Vincenzo Riccio vincenzo.riccio@unina.it
Tutor: Annarita Fasolino annarita.fasolino@unina.it
Cycle XXXI

	Credits year 1								Credits year 2								Credits year 3									
		1	2	3	4	5	6			1	2	3	4	5	6			1	2	3	4	5	6			
	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Total	Check
Modules	20	0	6	3	8	0	0	**17**	15							**0**	**0**							**0**	**17**	**30-70**
Seminars	10	6,8	1,4	0,7	0,8	0	0,5	**10,2**	5							**0**	**0**							**0**	**10,2**	**10-30**
Research	30	4	4	5	3	8	10	**34**	45							**0**	60							**0**	**34**	**80-140**
	60	10,8	11,4	8,7	11,8	8	10,5	**61,2**	65	0	0	0	0	0	0	**0**	60	0	0	0	0	0	0	**0**	**61,2**	**180**

References

[1] T. Meservy and K. Fenstermacher, "Transforming software development: an mda road map," Computer, vol. 38, no. 9, pp. 52–58, Sept 2005.

[2] A. Baresel, M. Conrad, S. Sadeghipour, and J. Wegener, "The Interplay between Model Coverage and Code Coverage," in Conference On Computer Aided Systems Theory, 2003.

[3] D. Amalfitano, V. De Simone, A. R. Fasolino and V. Riccio, "Comparing Model Coverage and Code Coverage in Model Driven Testing: An Exploratory Study", 2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), Lincoln, NE, 2015, pp. 70-73. doi: 10.1109/ASEW.2015.1830.

[4] J. Zander, Z. Dai, I. Schieferdecker, and G. Din, "From u2tp models to executable tests with ttcn-3 - an approach to model driven testing " in Testing of Communicating Systems, ser. Lecture Notes in Computer Science, F. Khendek and R. Dssouli, Eds. Springer Berlin Heidelberg, 2005, vol. 3502, pp. 289–303.

[5] Muccini H, di Francesco A, Esposito P. Software testing of mobile applications: Challenges and future research directions. Automation of Software Test (AST), 2012 7th International Workshop on, IEEE: Zurich, Switzerland, 2012; 29–35, doi:10.1109/IWAST.2012.6228987.

[6] Issa A, Sillito J, Garousi V. Visual testing of graphical user interfaces: An exploratory study towards systematic definitions and approaches. Proceedings of the 2012 IEEE 14th International Symposium on Web Systems Evolution (WSE), WSE '12, IEEE Computer Society: Washington, DC, USA, 2012; 11–15, doi:10.1109/WSE.2012.6320526.

[7] Zaeem RN, Prasad MR, Khurshid S. Automated generation of oracles for testing user-interaction features of mobile apps. Proceedings of the 2014 IEEE International Conference on Software Testing, Verification, and Validation, ICST '14, IEEE Computer Society: Washington, DC, USA, 2014; 183–192, doi:10.1109/ICST.2014.31.

[8] Lelli V, Blouin A, Baudry B. Classifying and qualifying gui defects. 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), 2015; 1–10, doi:10.1109/ICST.2015.7102582.

[9] Android G. Android - What To Test 2015. URL http://goo.gl/AL22tJ.

[10] Microsoft. Quickstart: Screen orientation for Windows Phone 8 2016. URL https://msdn.microsoft. com/en-us/library/windows/apps/jj207002(v=vs.105).aspx.

[11] ISO/IEC/IEEE 24765:2010 - Systems and software engineering. Standard, International Organization for Standardization, Geneva, CH, Dec. 2010.

[12] J. Lyndsay. A positive view of negative testing. Technical report, Workroom Production Ltd., 2003.