MAP-I: Doctoral Program in Informatics

2015-16 Edition

Proposal of a Curriculum Unit in Technologies

Software Testing

Ana Paiva
João Pascoal Faria
Rui Maranhão
UPorto

Alcino Cunha
UMinho

July, 08, 2015

Abstract: This document describes a Ph.D. level course, corresponding to a curriculum unit credited with 5 ECTS. It corresponds to a joint UPorto-UMinho proposal for OPTI (option in technologies) in the joint MAP-I doctoral program in Informatics. It is presented the programmatic component, the lecturing team, and the plans for the coordination with curriculum units in other PhD programs.

A. Programmatic Component

1. Theme, Justification and Context

Motivation

Software is increasingly present in our daily life. The software controls 80% of the functions of military aircraft, motor controls, TVs, mobile phones, etc., and it moves daily one billion dollars in the financial market, and is crucial for several businesses. Thus, the quality of software is becoming increasingly important. But assuring the quality of increasingly complex software systems is a challenging task. Failures in software are all too common and have a huge economic impact. According to NSF, the economic impact of inadequate software testing is estimated to represent 0.6% of the USA's GNP.

In the software industry, software testing typically consumes 30% to 50% of the development effort and schedule. Yet, the quality of delivered software products is not satisfactory. Considering the typical defect density of 1 to 7 defects/KLOC in delivered software products, typically with millions of lines of code, leads to a number of defects in delivered products in the order of thousands of defects. Hence, it is of upmost importance to improve testing efficiency and effectiveness, to reduce testing costs and improve product quality, namely through test automation and automatic fault localization.

Software testing is recognized as a major knowledge area or subject of software engineering, both by the IEEE's Guide to the Software Engineering Body of Knowledge (SWEBOK) and the ACM Computing Classification System.

2. Objectives and Learning Outcomes

This course aims to cover, both from the foundational and the methodological point of view, the concepts and techniques for Software Testing.

The course is not intended as an introductory survey on Software Testing, but as an opportunity of exposing students to cutting-edge research topics in this area, although presented in a coherent and integrated way. It is placed at a similar level and covers overlapping material with advanced modules in doctoral programs at leading academic institutions.

Upon successful completion of this curricular unit, students should be able:

- to recognize and explain the economic and social importance of software quality in general, and testing in particular;
- to understand the benefits and challenges of automated software testing techniques, such as mode-based testing or randomized testing;
- to understand the benefits and limitations of automated software test generation techniques;
- to understand the impact of testing in the subsequent phase of debugging, namely the impact on automated software fault localization;
- to identify test related measures (efficiency, effectiveness, coverage, rho, etc.);
- to understand the principles and techniques for test suite selection, minimization and prioritization.

3. Course Contents

Our society is increasingly dependent on the correct functioning of software systems, so the software industry should strive to deliver essentially defect free software in a cost-effective way, by using more effective and efficient defect detection and prevention techniques than are in common use today. The goal of this unit is to cover some of the most promising techniques in that direction, namely concerning software testing. Testing is the main defect detection technique in software industry. Manual testing is infeasible because of the ever-growing size and complexity of software systems, so the main focus in this course will be placed in the study of test automation techniques, notably automatic test generation.

The course is organized in three units:

3.1 Software testing fundamentals

This unit will review basic concepts related to testing, namely:

- Test process;
- Test levels and test types;
- Test coverage analysis and code instrumentation;
- Advanced unit testing techniques;
- Test suite minimization and prioritization for regression testing;

- Test management;
- Metrics for assessing test quality.

3.2 Automated software testing

This is the core unit of the course, focusing on the principles and techniques for automated software testing. A key topic to be addressed in this unit is model-based testing, namely the usage of formal specification languages to build models of software to serve as oracles and model finding techniques to automatically generate test cases.

More specifically, this unit will cover the following topics:

- Survey of automated testing techniques;
- Survey of model-based testing techniques (from patterns, algebraic specifications, state-based specifications, etc.);
- Automatic GUI testing;
- Test case generation: theory, techniques and tools;
- Specifying formal models of software with Alloy to support model-based testing;
- Using model finders to automate test generation;
- Automatic randomized testing with QuickCheck-like techniques.

3.3 Life after Testing: Debugging

Software reliability/quality can generally be improved through extensive testing and debugging, but this is often in conflict with market conditions: software cannot be tested exhaustively, and of the bugs that are found, only those with the highest impact on the user-perceived reliability can be solved before the release. In this typical scenario, testing reveals more bugs than can be solved, and debugging is the bottleneck for improving software reliability/quality. Automated debugging techniques can help to reduce this bottleneck. These techniques give a diagnosis for failures that are detected during the execution of a program, which can help programmers to locate their root causes, and thus to reduce the effort spent on manual debugging.

The unit is structured into the following topics:

- Software evolution/transformation and semi-automatic test case co-adaptation;
- (Semi-) automatic fault localization;
- Devise proper test suites which help in the subsequent debugging phase;
- Introducing to test sequencing for improving the time spent on debugging.

4. Teaching Methods and Student Assessment

Studies show the need to consider software testing as a multidisciplinary activity that requires systematization. This course will allow:

- learning the fundamental concepts and principles of software testing;
- knowing and understanding the solutions and proven practices to improve software testing through exploration of examples and case studies;

- applying the knowledge gained by using and adapting known solutions for a particular problem in an individual project.

No textbook adequately covers the course's range of topics, so a diversity of bibliographic elements (books, journals and conference proceedings) will be used.

Classes

The class meetings are meant to be conversational, and we encourage students to ask questions and make comments. Consequently, the discussion may follow tangents to the prepared lecture, but they should be fruitful, informative, and thought provoking. These classes are conducted by all the elements of the lecturing team.

Panels and talks

Two panels or talks will be organized to complement the topics covered by the formal classes. These panels or talks will involve both the lecturing team and possibly faculty members that are not formally associated with this UC.

Readings

Each week, the students must read papers or some few supplemental materials provided, related to the topics discussed in class. All reading assignments come from journals and conference proceedings. This exposes many students to extensive readings from the research literature for the first time. To help them with their reading, we require them to write a brief summary for each paper, and submit it electronically before the next class. We also ask them to submit a list of questions about the readings, which we try to work into the lecture if possible. Whenever applicable, students will also be encouraged to experiment with the technologies mentioned in the reading materials. During the last few weeks of the course, we no longer require reading summaries, to give students more time to focus on the project.

Individual research project

Whilst the goal of the readings is to develop a critical but shallow view over a broad range of topics addressed in classes, the goal of the individual research project is to develop a more in depth understanding on a software testing research topic, matching as much as possible each student interests, as well as the lecturers areas of expertise. Projects are designed to combine a state of the art analyses with an experimental assessment. Grading of individual research projects is based on an oral presentation (for a more methodological project) or demonstration (for a more technological project), and a final written report. No two students can work on the same project. A few weeks into the course, descriptions of possible projects are handed out to students, who are also encouraged to propose projects of their own. Once students complete their project, they must demonstrate it, make an oral presentation, and submit a final written report.

5. Basic Bibliographic References

- Ilene Burnstein. *Practical software testing*, Springer, 2003.
- Aditya P. Mathur. *Foundations of software testing*, Pearson Education India, 2008.
- M. Utting, M. Legeard. *Practical Model-Based Testing: A Tools Approach*, Morgan Kaufmann, 2007.

- R. M. Poston. *Automating Specification-based Software Testing*, IEEE Press, 1996.
- Software Verification and Analysis: An Integrated, Hands-On Approach, by Janusz Laski and William Stanley (Nov 5, 2010).
- Christof Ebert, Reiner Dumke. Software Measurement: Establish - Extract - Evaluate – Execute. Springer-Verlag, 2007.
- Daniel Jackson. *Software Abstractions – Logic, Language, and Analysis (revised edition)*. MIT Press, 2011.
- Andreas Zeller.*Why Programs Fail: A Guide to Systematic Debugging*. Elsvier, 2009.

B. Lecturing Team

1. Team Presentation

This course is supported by a team involving researchers from the University of Porto (Ana Paiva, João Pascoal Faria and Rui Maranhão) and the University of Minho (Alcino Cunha).

All team members are working, and have worked actively in the past few years, on topics that are directly related to the subjects covered by this course, as detailed below.

2. Coordinator

The coordinator of the unit is Ana Paiva.

3. Short Presentation of Team Members

In the sequel we introduce a brief presentation of each team member, which includes, for each of them, up to 5 key publications related to the scientific area in which this course is proposed. All CVs are supplied in separate PDF documents.

Ana Paiva is assistant Professor at the Informatics Engineering Department of the Faculty of Engineering of University of Porto (FEUP) where she works since 1999. She teaches subjects like Software Testing, Formal Methods and Software Engineering, among others. She belongs to the group on Software Engineering (softeng.fe.up.pt) which gathers researchers and post graduate students with common interests in software engineering. She has a PhD in Electrical and Computer Engineering from FEUP with a thesis titled "Automated Specification Based Testing of Graphical User Interfaces". Her expertise is on the implementation and automation of the model based testing process. She has been developing research work in collaboration with Foundation of Software Engineering research group within Microsoft Research where she had the opportunity to extend Microsoft's model-based testing tool, Spec Explorer, for GUI testing. She was a member of the CYTED network on software verification and validation (REVVIS), she is vice-president of the PSTQB (Portuguese Software Testing Qualification Board) board (www.pstqb.pt), member of the Council of the Department of Informatics Engineering, and member of the Council of Representatives of FEUP.

Key Publications:

- Faria, J. P., & Paiva, A. C. R. (2014). A Toolset for Conformance Testing against UML Sequence Diagrams based on Event-Driven Colored Petri Nets. International Journal on Software Tools for Technology Transfer, December 2014, 1-20. doi: 10.1007/s10009-014-0354-x
- Moreira, R. M., & Paiva, A. C. (2014). PBGT Tool: An Integrated Modeling and Testing Environment for Pattern-based GUI Testing. Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE'14) (pp. 863-866). Vasteras, Sweden: ACM. doi:10.1145/2642937.2648618
- Moreira, R. M., Paiva, A. C., & Memon, A. (2013). A pattern-based approach for GUI modeling and testing. IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), (pp. 288-297). doi:10.1109/ISSRE.2013.6698881
- Specification-driven Unit Test Generation for Java Generic Classes , in iFM - 9th International Conference on Integrated Formal Methods, Francisco R. de Andrade, João P. Faria, Antónia Lopes, Ana C. R. Paiva, Pisa, Italy, June 18-21, 2012.
- Test case generation from mutated task models, in the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'11), Ana Barbosa, Ana C. R. Paiva, José Creissac Campos, Pisa, Italy - June 13-16, 2011.

João Pascoal de Faria is assistant professor at the Department of Informatics Engineering of Faculty of Engineering of University of Porto (FEUP) and researcher at INESC TEC. He was vice-president of the Sectorial Commission for the Quality in Information and Communication Technologies (CS/03) in the scope of the Portuguese Quality Institute (IPQ) between 2010 and 2014. In the past, he also worked with several software companies (Novabase, Sidereus, and Medidata), and was co-founder of two other (Qualisoft and Strongstep). He has more than 20 years of experience in teaching, research and consultancy in several software engineering areas. He is the main author of a rapid application development tool (SAGA), based on domain specific languages, that recently completed 25 years of continuous market presence and evolution. He is currently involved in research projects and supervisions in the areas of model-based testing, software process improvement, and model-driven development.

Key Publications:

- Faria, J. P., & Paiva, A. C. R. (2014). A Toolset for Conformance Testing against UML Sequence Diagrams based on Event-Driven Colored Petri Nets. *International Journal on Software Tools for Technology Transfer*, December 2014, 1-20. doi: 10.1007/s10009-014-0354-x
- Faria, J. P., Lima, B., Sousa, T. B., & Martins, A. (2014). A Testing and Certification Methodology for an Open Ambient-Assisted Living Ecosystem. *International Journal of E-Health and Medical Communications*, Volume 5, Issue 4, 90-107. doi: 10.4018/ijehmc.2014100106
- Lima, B., & Faria, J. P. (2015). An Approach for Automated Scenario-based Testing of Distributed and Heterogeneous Systems. *10th International Conference on Software Engineering and Applications (ICSOFT-EA 2015),* Colmar, France *(accepted for publication)*
- Raza, M., & Faria, J. P. (2014). A Benchmark-based Approach for Ranking Root Causes of Performance Problems in Software Development, *The 15th International Conference on Product-Focused Software Process Improvement (PROFES'2014)*, Helsinki, Finland.
- Cruz, A. M. R., & Faria, J. P. (2010). A Metamodel-based Approach For Automatic User Interface Generation. In D. C. Petriu, N. Rouquette, & Ø. Haugen (Eds.), *Lecture Notes in Computer Science*: Vol. 6394. *MODELS'10 Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems* (pp. 256-270), Berlin, Germany: Springer-Verlag. doi:10.1007/978-3-642-16145-2_18 *(best paper award)*

Alcino Cunha is assistant professor at the Department of Informatics of Universidade do Minho, and member of the Software Engineering research group of the High-Assurance Software Laboratory of INESC TEC. In recent years, his research is focused on the topic of automated software engineering, namely developing new techniques and tools for model repair and (bidirectional) transformation, scenario exploration, model checking, and for the integration of formal specification and analysis techniques, namely Alloy, in the standard MDA software design methodology. Related to these topics, he has collaborated with several international research teams, namely the Software Design Group at MIT, the group that developed the Alloy formal modeling language, or the Modeling and Information Processing group at ONERA, the French Aerospace Lab. Since 2007 he has also taught a course on formal specification and analysis with Alloy at the informatics master degree at Universidade do Minho.

Key publications:

- Alcino Cunha, Ana Gabriela Garis, Daniel Riesco: Translating between Alloy specifications and UML class diagrams annotated with OCL. Software and System Modeling 14(1): 5-25 (2015)
- Nuno Macedo, Alcino Cunha, Tiago Guimarães: Exploring Scenario Exploration. FASE 2015: 301-315
- Alcino Cunha, Nuno Macedo, Tiago Guimarães: Target Oriented Relational Model Finding. FASE 2014: 17-31
- Nuno Macedo, Alcino Cunha: Implementing QVT-R Bidirectional Model Transformations Using Alloy. FASE 2013: 297-311
- Nuno Macedo, Tiago Guimarães, Alcino Cunha: Model repair and transformation with Echo. ASE 2013: 694-697

Rui Maranhão (publishes as Rui Abreu) graduated in Systems and Computer Engineering from University of Minho, Portugal, carrying out his graduation thesis project at Siemens S.A., Portugal. Between September 2002 and February 2003, Rui followed courses of the Software Technology Master Course at University of Utrecht, the Netherlands, as an Erasmus Exchange Student. He was an intern researcher at Philips Research Labs, the Netherlands, between October 2004 and June 2005. He received his Ph.D. degree from the Delft University of Technology, the Netherlands, in November 2009. During his PhD studies period (2005-2009) he was also a research associate at the Embedded Systems Institute, the Netherlands. Currently, he is an assistant professor at the Faculty of Engineering of University of Porto, Portugal.

Key publications:

- Campos, J., Arcuri, A., Fraser, G., & Abreu, R. (2014, September). Continuous test generation: enhancing continuous integration with automated test generation. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (pp. 55-66). ACM.
- Campos, J., Abreu, R., Fraser, G., & d'Amorim, M. (2013, November). Entropy-based test generation for improved fault localization. In Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on (pp. 257-267). IEEE.
- Gouveia, C., Campos, J., & Abreu, R. (2013, September). Using HTML5 visualizations in software fault localization. In Software Visualization (VISSOFT), 2013 First IEEE Working Conference on (pp. 1-10). IEEE.

- Hofer, B., Riboira, A., Wotawa, F., Abreu, R., & Getzner, E. (2013). On the empirical evaluation of fault localization techniques for spreadsheets. InFundamental Approaches to Software Engineering (pp. 68-82). Springer Berlin Heidelberg.
- Abreu, R., Zoeteweij, P., & Van Gemund, A. J. (2009, November). Spectrum-based multiple fault localization. In Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on (pp. 88-99). IEEE.