CS Curricula of the Most Relevant Universities in Brazil and Abroad: Perspective of Software Testing Education

Pedro Henrique Dias Valle, Ellen Francine Barbosa, José Carlos Maldonado
Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP)
São Carlos/SP, Brazil, 13560-970
Email: pedrohenriquevalle@usp.br, francine@icmc.usp.br, jcmaldon@icmc.usp.br

***Abstract*—Software testing is a relevant activity to provide evidencies of the quality of software products. However, few courses in the computing area provide an adequate body of knowledge to the students and few of them pursue the software development practices and activities related to VV&T (Verification, Validation, and Testing), specifically testing, leading to a recognized lack of professionals that master the underlying VV&T concepts and principles. In this paper, we provide an overview concerning the software testing education in the most relevant brazilian universities and abroad. We analyze the reference curricula proposed by SBC (Brazilian Computer Society) and ACM (Association for Computing Machinery) for undergraduate computer courses as well as the curricula of the most relevant universities in Brazil and abroad. We identify the effort allocated, the main approaches and the main tendencies to facilitate the teaching of software testing. It is clear that there is a need to integrate software testing education with other disciplines along the CS undergraduate courses.**

Keywords—Software Testing, Curricula Recommendations, Integrated Education.

I. Introduction

Software testing aims at executing programs or models with specific inputs analyzing if these products behave as expected, leading to a detailed analysis of the software with the goal to identify failures and after, through debugging, to eliminate the faults that originate the failures. The software testing is a dynamic activity since it is based on the execution of programs or other executable intermediate artifacts. [1]. Software testing involves four phases: test planning, test case design, test execution and evaluation of testing results. These phases must be planned and applyed along the software development processes, according to previously established testing strategies [2].

Despite the software testing being recognized as a relevant activity in quality assurance of software products, the test industry has found a lack of specialized professionals in this area [3]. This fact can be related with the difficulty and inefficiency in software testing education, since there is a lack of well established environments and methods that motivate the students to learn contents related to the software testing activity [4].

To know how the testing professionals are trained, we analyzed the reference curricula proposed by Brazilian Computer Society (SBC) and by Association for Computing Machinery (ACM) for undergraduate courses in Computer Science. The goal of the reference curricula is to provide support and guidelines for the definition, implementation and evaluation of undergraduate courses in Computer Science. These curricula recommend that the software testing contents be addressed as one of the topics of Software Engineering course, called Verification, Validation and Testing Software (VV&T) [5, 6, 7, 8, 9].

The SBC and ACM recommendations, however, do not explicitly motivate to provide the students an integrated view of software testing contents with other disciplines and practices related to the development of software products, such as: Algorithms, Object-Oriented Programming, Data Structures, Analysis and Requirements Specification, Software Reuse, among others. As a consequence, the students do not acquire a "test culture" adequate for software product development, since they do not embody the habit of rigorously testing their programs from the moment they learn to program. It is clear that it is indispensable to revisit the methodology of software testing education integrating the software testing contents with other disciplines along the CS undergraduate courses.

The aim of this paper is to characterize how software testing education has been approached in the CS undergraduate courses at universities in Brazil and abroad. We identify the effort allocated, the main approaches and the main tendencies to facilitate the software testing education.

This paper is organized as follows: In Section II it is presented the research methodology used for the selection of universities that had their CS reference curricula analyzed, as well as the reference curricula proposed by SBC and ACM. In Section III, we present an overview about software testing education, providing an analysis of the reference curricula proposed by SBC and ACM and of the CS curricula of the most relevant universities in the world. In Section IV, we present the main approaches identified to support software testing education. In Section V, we discuss the main tendencies in teaching of software education. Finally, in Section VI we present the conclusions and future work.

II. METHODOLOGY

For the selection of the brazilian universities that had their curricula analyzed we used the ranking made available by

the RUF (Ranking Universitário Folha)[1] of the year 2014. The RUF is an annual evaluation about the higher teaching of the Brazil accomplished by the newspaper Folha de São Paulo. In this evaluation are ranked the most relevant brazilian universities, including publics and privates, by analyzing five indicators: research, internationalization, innovation, education and marketing.

For the selection of the most relevant universities in the world, we used the ranking 2014-2015 made available by Times Higher Education[2], that is an english magazine about articles and news of the higher education. This magazine provides annually a ranking of the most relevant universities in the world.

Finally, we analyzed the reference curricula made available by SBC[3] and ACM[4] for CS undergraduate courses: Computer Science, Computer Engineering, Information Systems, Software Engineering, among others. We conducted a systematic mapping to identify the main tendencies for teaching software testing, aiming at contributing to the reformulation of current practices [10].

III. DIAGNOSIS OF THE SOFTWARE TESTING EDUCATION

Analysing the curricula of the selected national and international universities, we obtained an diagnosis of how software testing professionals are educated and trained along their undergrad activities. In addition, we perform an analysis of the reference curricula proposed by SBC and ACM. We summarize the teaching practices of software testing at the selected universities in Brazil and abroad.

A. Brazilian Scenario

The reference curricula proposed by SBC for CS undergraduate courses suggests that the contents related to the software testing could be approached in one of the topics of the Software Engineering discipline, called Verification, Validation and Testing (VV&T), in which few lectures are dedicated to testing [5, 11, 12]. This recommendation has induced a practice in which the contents are addressed isolated with few integration with other disciplines.

We analyzed the curricula of the courses in Computer Science, Computer Engineering, Information Systems and Software Engineering of the 25 most relevant universities in Brazil according to the *ranking* made available by RUF in 2014. In Table I we present the brazilian universities that had their curricula analyzed.

In general, the brazilian universities address software testing education in the scope of the Software Engineering discipline, in the topic Verification, Validation and Testing (VV&T) without integration with other disciplines. In this case, usually, less than ten hours are assigned to the teaching of software testing in these courses, i.e., there is no time enough to adequately address a software testing practice in the CS undergrad courses

TABLE I. THE MOST RELEVANT BRAZILIAN UNIVERSITIES

Brazilian Universities	**Abbreviation**
University of Sao Paulo	USP
Federal University of Minas Gerais	UFMG
Federal University of Rio de Janeiro	UFRJ
Federal University of Rio Grande do Sul	UFRGS
State University of Campinas	UNICAMP
São Paulo State University	UNESP
Federal University of Santa Catarina	UFSC
University of Brasilia	UNB
Federal University of Paraná	UFPR
Federal University of São Carlos	UFSCAR
Federal University of Pernambuco	UFPE
Federal University of São Paulo	UNIFESP
Federal University of Ceara	UFC
Federal University of Bahia	UFBA
Federal University of Santa Maria	UFSM
Federal University of Fluminense	UFF
State University of Rio de Janeiro	UERJ
Pontifical Catholic University of Rio Grande do Sul	PUCRS
Federal University of Viçosa	UFV
Pontifical Catholic University of Rio de Janeiro	PUC-RIO
Federal University of Rio Grande do Norte	UFRN
Federal University of Goias	UFG
State University of Maringa	UEM
State University of Londrina	UEL
Federal University of Paraiba	UFPB

There are also few brazilian universities that have a specific and compulsory discipline for software testing,

[1] Available at: http://ruf.folha.uol.com.br

[2] Available at: http://www.timeshighereducation.co.uk/

[3] Avaliable at: http://www.sbc.org.br/

[4] Avaliable at: http://www.acm.org/education/curricula-recommendations

such as: University of Brasilia (UNB), Federal University of Ceará (UFC), Pontifical Catholic University of Rio Grande do Sul (PUCRS), Federal University of Rio Grande do Norte (UFRN) and Federal University of Goiás (UFG). Few other universities have an elective discipline for software testing, such as: Federal University of Rio de Janeiro (UFRJ) and Institute of Mathematics and Computer Science of University of São Paulo (ICMC/USP).

However, it should be observed that the contents of software testing are methodologically addressed without providing the students an integrated view of other relevant disciplines for software product development, such as: Algorithms, Object-Oriented Programming, Data Structures, Analysis and Requirements Specification, Software Reuse, Data Base, among others.

B. *International Scenario*

ACM has also proposed reference curricula for CS undergraduate courses: Computer Science, Software Engineering, Computer Engineering and Information Systems. ACM suggests that contents related to the software testing could be approached in topics of disciplines such as Software Engineering, Verification and Validation of Software, Computer Systems Engineering and Programming Fundamentals, inducing more integrated practices of testing with other disciplines. However, like in the brazilian universities, less than ten hours are assigned to the teaching of software testing in these courses, i.e., there is no time enough to adequately address a software testing practice in the CS undergrad courses [13, 7, 9].

We also performed a search in the CS curricula of the most relevant universities in the world with the goal to identify how the contents related to the software testing have been addressed. In Table II we present the international universities that had their curricula analyzed.

Table II. THE MOST RELEVANT UNIVERSITIES IN THE WORLD OF COMPUTER SCIENCE AREA

International Universities	Country
Massachusetts Institute of Technology	United States
Stanford University	United States
California Institute of Technology	United States
Princeton University	United States
University of Cambridge	United Kingdom
Imperial College London	United Kingdom
University of Oxford	United Kingdom
Swiss Federal Institute of Technology Zürich	Switzerland
University of California - Los Angeles	United States
University of California - Berkeley	United States
Georgia Institute of Technology	United States
École Polytechnique Fédérale de Lausanne	Switzerland
National University of Singapore	Singapore
Carnegie Mellon University	United States
Cornell University	United States
Northwestern University	United States
University of Illinois at Urbana Champaign	United States Delft
University of Technology	Netherlands Hong Kong
University of Science and Technology	Japan Harvard
University	United States
University of California - Santa Barbara	United States

In the curricula of these universities, the contents related to the software testing are considered in the topics of the disciplines of Software Engineering and/or Programming, as proposed by ACM. The software testing contents are addressed isolatedly, without a relationship with other disciplines during the CS undergraduate courses.

It can be observed, with few exceptions, that in the CS courses, either in Brazil or abroad, software testing is addressed only in the disciplines of Software Engineering and/or Programming Fundamentals. Few lectures are assigned to the teaching of software testing, providing to students only an overview of the contents of this area. In addition, software testing education is approached only at the end of the undergraduate courses. Therefore, the software testing professionals, in general, do not receive a good education at the universities to start working inside companies.

It would be interesting to integrate software testing education with other disciplines along the CS undergraduate courses, with projects and tasks that would lead the students to explore in an integrated perspective the skills and competences developed in the course, including software testing. In this direction, it would be mandatory to revisit and modify the reference curricula proposed by SBC and ACM, since they provide crucial directions and elements for the definition, implementation and evaluation of CS undergraduate courses. There is the need to induce and motivate software testing contents be addressed in an integrated perspective with other disciplines along the CS undergraduate course, making evident to students the importance of testing their programs (products) before being released. Consequently, the developers would create the habit of testing their programs since they start learning how to program. This would lead to he creation of an adequate "test culture" adequate among students and professionals.

IV. Approaches to support Software Testing Education

In this perspective, the need to improve software testing education is more than evident. It is indispensable to develop environments and tools to facilitate the teaching and training of software testing, providing to the students and professionals skills and motivation to work with software testing [14]. Recently, different approaches, methodologies and tools have been used to facilitate software testing education in an integrated approach along CS undergraduate courses.

To characterize the main initiatives used in software testing education, we carried out a systematic mapping [10]. In this study we identified 11 different initiatives to support software testing education: Educational Modules, Peer Review, Educational Games, Quiz, Tutorial, Test Driven Development (TDD), Problem-Based Learning, Integrated Teaching of Software Testing and Programming, Social Network, Software Residence and Based-Learning Performance. Following, we present a description of the four most used of these initiatives.

- **Educational Modules:** They are concise units of study, composed of theoretical contents combined with practical activities and evaluations, supported by technological and computing resources [15];
- **Educational Games:** They are games that facilitate the learning of concepts and ideas, providing attractive educational practices, in which the users can learn more actively, dynamic and motivated, allowing to acquire knowledge combined with the fun [16];
- **Test Driven Development (TDD):** It is a software development technique in which the developers write automated test cases for each functionality. Then, code is produced that can be validated by the previously written tests [17] and;
- **Integrated Teaching of Software Testing and Programming:** It is the teaching of basic concepts of programming together with the software testing concepts. Several studies have shown that the integrated teaching of these topics provides benefits, developing the skills of analysis and understanding in the students [18, 19, 20].

Educational games and the integrated teaching of software testing and programming are the two most used initiatives identified to support the software testing education. We perceived that these initiatives can be considered as the main tendencies in software testing education. These initiatives are briefly discussed next.

V. Tendencies of Software Testing Education

Every software product must be properly tested before being released to the user. Despite the software testing being recognized as an important activity in the quality assurance of software products, many students feel unmotivated to learn the contents related to the software testing. In addition, many software testing professionals are devalued because, in general, their salaries are lower compared with other positions in the software development companies [20, 4].

However, this scenario has been changed and some companies have changed their allocation strategies and accountability of these professionals in the development teams, once that they perceived that these activities are entirely related. Although there is no scientific evidence, researchers say that testers with good programming skills perform tests with better efficiency [3]. Thus, many companies are now understanding that software testing is an important activity for the development of high quality software products.

As mentioned before, it is fundamental to develop environments and tools that facilitate the teaching and training of software testing, providing to the students and professionals training and motivation to work with software testing [14]. To address these problems, software testing tools and environments have been used to facilitate the software test execution process, allowing to detect the greater number of defects so that it can provide evidences of the quality of software products. Currently, there are several software testing tools, such as: JaBUTi, muJava, SPACES, among others [21, 22, 23]. In the website of the Núcleo de Apoio à Pesquisa in Software Livre (NAPSoL)[5] some open source software testing tools are available. Such tools can be used to support the software testing education.

However, the isolated adoption of these tools is not enough to motivate and facilitate software testing education. It is also necessary to use different methodologies and to approach the testing contents and concepts in an integrated perspective along the CS undergraduate courses. Following, we present the two most relevant initiatives in this direction: the integrated perspective of software testing education and educational games.

C. *Integrated Perspective of Software Testing Education Contents*

The integrated teaching of software testing and programming can help the development of skills of analysis and comprehension in the process of training students, because for testing is necessary that the students know the behavior and semantics of their programs [17, 3]. We present some of the initiatives in this direction.

[5] Available at: http://napsol.icmc.usp.br/

The paper proposed by Corte and Maldonado [20] had the goal to provide mechanisms for integrating the programming fundamentals teaching with software testing education. For this, they developed an integrated educational module of object-oriented programming and software testing. This module divided the contents of programming fundamentals, identifying which the software testing contents could be applied to the programming contents. Thus, since the beginning of the CS undergraduate course, students are exposed to an integrated perspective exploring the contents and fundamentals of programming and software testing.

Code Hunt [24] is a game that also explores the software testing education with programming. In this game, the players must find a code fragment observing the expected result for the code fragment uncovered. The Code Hunt offers to the user a faulty code and some tips on what should be the result of the program when it is correct. For the code to be considered correct, the player must use several test cases [25].

As the code fragment is written the test cases change, enabling that the players obtain new experiences. The game Code Hunt teaches programming using software testing techniques. This technique is known like Test Driven Development (TDD), in which developers guided test cases produce code that can be validated by the tests [24].

Others initiatives have been proposed for the integration of software testing contents with others disciplines. Among them, we find the tool called ProgTest, which is an environment for the submission and evaluation of practical exercises based on the software testing activities. The ProgTest environment [3] evaluates practical exercises written in programming languages Java and C. For the evaluation of the exercises, this tool uses the functional, structural and fault-based test criteria. By using this tool, the teachers can automate the process of evaluating the codes and the test case sets that, in general, is carried out manually by the teacher. [3].

Web-CAT also is a tool that was developed to support the software testing education with programming [3]. This tool have the goal to support the submission and evaluation of practical exercises of programming assisting the test driven development [26]. Web-CAT evaluates the exercises that was submitted analyzing the following criteria: i) readability/project, analyzed manually by an assistant teacher;
ii) style/codification, analyzed automatically by static analysis tools; and iii) test/correction, analyzed automatically by software testing tools, among the programming languages included, there are Java, C, C++ and Pascal [26].

Marmoset is an environment for the submission of programming practical exercises based on software testing activities. The main goal of this environment was to improve the students learning in relation to programming contents and to know how students learn to program. Test cases used to test the programs submitted to Marmoset can be of four types:
i) student test, provided by the students; ii) public testing, provided to students before they begin their exercises; iii) release test, made available by teachers in specific conditions or after the end of the deadline; and iv) secret tests, they are written by the teacher and made available only at the end of the deadline of the exercises [27].

D. Educational Games to Support the Software Testing Education

Educational games raised the interest of educational technology researchers because there was a large increase of the video game industry in recent years. In addition, many researchers argue that these tools can facilitate the learning of concepts and ideas, because these games motivate and provide pleasures to the players while they play and learn. Therefore, we perceived that the educational games allow their users obtaining knowledge combined with the fun [16]. In this context, some educational games have been proposed to support software testing education, for instance: iTest Learning [16], Jogo das 7 Falhas [28], JETS [29], U-TEST [30], iLearnTest [31], among others.

However, the educational games previously refered to, dealt only with the contents of functional test and do not have an integrated teaching methodology with other courses. Therefore, the students do not have an integrated vision of software testing concepts with other disciplines, compromising their skills. This fact have contributed to the lack of an adequate "test culture" to the development of software products, since they did not have the habit of adequately testing their first programs.

This perspective has motivated us to develop an educational game that would address the teaching of software testing together with the teaching of programming fundamentals. The idea is to provide the students an integrated teaching of software testing contents with other disciplines of the Computer Science curricula. Thus, the students would have a better perspective of integrating testing in the scope of software product development. This would improve the students "test culture". The game will address the functional test, structural test and fault-based test, providing a more motivating environment to the students.

VI. Conclusions and Future Work

In this paper, we identified how software testing education has been approached in CS undergraduate courses in the most relevant universities in Brazil and abroad. We performed an analysis in the CS reference curricula proposed by SBC and ACM as well as in the CS curricula of the most relevant universities in Brazil and in the world. We observed that, in general, software testing contents are not addressed in an integrated perspective with other disciplines. It can be observed, with few exceptions, that in the CS courses, either in Brazil or abroad, software testing is addressed only in the disciplines of Software Engineering and/or Programming Fundamentals. Few lectures are assigned to the teaching of software testing, providing to students only an overview of the contents of this area.

We identified 11 different approaches to support software testing education, among them: Educational Games, Integrated Teaching of Software Testing Teaching and Programming, Educational Modules and Test Driven Development. In this paper we presented the two main tendencies in software testing education: i) integration of software testing contents teaching with other disciplines along the CS undergraduate courses; and
ii) use of educational games in software testing education. We identified tools and games for supporting these tendencies of software testing education.

It should be highlighted that it would be interesting to integrate software testing education with other disciplines along the CS undergraduate courses, with projects and tasks that would lead the students to explore in an integrated perspective the skills and competences developed in the course, including software testing. In this direction, it would be mandatory to revisit and modify the reference curricula proposed by SBC and ACM, since they are provide crucial directions and elements for the definition, implementation and evaluation of CS undergraduate courses.

Since software testing is considered a difficult discipline to be taught only by lectures and theoretical classes, the development of tools and environments to facilitate software testing education is mandatory. This scenario motivated the development of an educational game that will address the software testing education along with programming teaching, combining the two approaches more used to support the software testing education: Educational Games and Integrated Teaching of Software Testing and Programming.

REFERENCES

[1] M. E. Delamaro, J. C. Maldonado, and M. Jino,*Introdução ao Teste de Software*. Elsevier, 2007.

[2] R. Pressman, *Software Engineering: A Practitioner´s Approach*. McGraw Hill, Inc., 2010.

[3] D. M. de Souza, J. C. Maldonado, and E. F. Barbosa, "Aspectos de desenvolvimento e evolução de um ambiente de apoio ao ensino de programação e teste de software," in *Anais do Simpósio Brasileiro de Informática na Educação*, 2012.

[4] J. Smith, J. Tessler, E. Kramer, and C. Lin, "Using peer review to teach software testing," in *Proceedings of the ninth annual international conference on International computing education research*. ACM, 2012.

[5] SBC, "Currículo de referência para cursos de graduação em bacharelado em ciência da computação e engenharia de computação," in *Sociedade Brasileira deComputação*, 2005.

[6] L. Cassel, A. Clements, G. Davies, M. Guzdial, R. McCauley, A. McGettrick, B. Sloan, L. Snyder, P. Tymann, and B. W. Weide, "Computer science curriculum 2008: An interim revision of cs 2001," in *Association for Computing Machinery*, 2008.

[7] R. J. LeBlanc, A. Sobel, J. L. Diaz-Herrera, T. B. Hilburn *et al.*, *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. IEEE Computer Society, 2006.

[8] C. Mao, "Towards a question-driven teaching method for software testing course," in *International Conference on Computer Science and Software Engineering*. IEEE, 2008.

[9] H. Topi, J. S. Valacich, R. T. Wright, K. Kaiser, J. F. Nunamaker Jr, J. C. Sipior, and G.-J. de Vreede, "Is 2010: Curriculum guidelines for undergraduate degree programs in information systems," *Communications of the Association for Information Systems*, 2010.

[10] P. H. D. Valle, E. F. Barbosa, and J. C. Maldonado, "Um mapeamento sistemático sobre ensino de teste de software," in *Anais do XXVI Simpósio Brasileiro de Informática na Educação*, 2015.

[11] SBC, "Currículo de referência para cursos de licenciatura em computação," in *Sociedade Brasileira de Computação*, 2002.

[12] SBC., "Currículo de referência para cursos de graduação em bacharelado em sistemas de informação," in *Sociedade Brasileira de Computação*, 2003.

[13] ACM, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, 2013.

[14] W. Wong, A. Bertolino, V. Debroy, A. Mathur, J. Offutt, and M. Vouk, "Teaching software testing: Experiences, lessons learned and the path forward," in *Conference on Software Engineering Education and Training*, 2011.

[15] E. Barbosa and J. Maldonado, *E-Infrastructures and Technologies for Lifelong Learning: Next Generation Environments*, 2011, ch. Collaborative development of educational modules: a need for lifelong learning.

[16] V. Farias, C. Moreira, E. Coutinho, and I. S. Santos, "itest learning: Um jogo para o ensino do

planejamento de testes de software." in *V Fórum de Educação em Engenharia de Software*. Natal: Simpósio Brasileiro de Engenharia de Software, 2012.

[17] S. H. Edwards, "Teaching software testing: Automatic grading meets test-first coding," in *Conference on Object-oriented Programming, Systems, Languages, and Applications*. ACM, 2003.

[18] D. M. de Souza, M. H. da Silva Batista, and E. F. Barbosa, "Avaliação de qualidade de um ambiente de apoio ao ensino de programação," *Revista Novas Tecnologias na Educação*, vol. 12, 2014.

[19] E. Barbosa, M. Silva, C. Corte, and J. Maldonado, "Integrated teaching of programming foundations and software testing," in *Frontiers in Education Conference*, 2008.

[20] C. K. D. Corte and J. C. Maldonado, "Ensino integrado de fundamentos de programação e teste de software," Master's thesis, Universidade de São Paulo. Instituto de Ciências Matemáticas e de Computação, 2006.

[21] A. Vincenzi, W. Wong, M. Delamaro, and J. Maldonado, "Jabuti: A coverage analysis tool for java programs," in *Simpósio Brasileiro de Engenharia de Software*, 2003.

[22] Y.-S. Ma, J. Offutt, and Y. R. Kwon, "Mujava: An automated class mutation system: Research articles," *Software Testing, Verification and Reliability*, 2005.

[23] D. Barbosa, W. Andrade, P. Machado, and J. Figueiredo, "Spaces–uma ferramenta para teste funcional de componentes," in *Simpósio Brasileiro de Engenharia de Software*, 2004.

[24] N. Tillmann, J. Bishop, N. Horspool, D. Perelman, and T. Xie, "Code hunt - searching for secret code for fun," *Proceedings of the 7th International Workshop on Search-Based Software Testing*, June 2014.

[25] N. Tillmann, P. de Halleux, T. Xie, and J. Bishop, "Code hunt: Gamifying teaching and learning of computer science at scale," *Conference on Learning at Scale*, 2014.

[26] S. H. Edwards and M. A. Perez-Quinones, "Web-cat: automatically grading programming assignments," in *Conference on Innovation and Technology in Computer Science Education*. ACM, 2008.

[27] J. Spacco, W. Pugh, N. Ayewah, and D. Hovemeyer, "The marmoset project: an automated snapshot, submission, and testing system," in *Symposium on Object-oriented programming systems, languages, and applications*. ACM, 2006.

[28] L. L. Diniz and R. L. S. Dazzi, "Jogo das sete falhas: Um jogo educacional para apoio ao ensino do teste caixa preta," in *Anais do Computer on the Beach*, 2011.

[29] T. G. da Silva and F. M. Muller, "Jogos sérios em mundos virtuais: uma abordagem para o ensino-aprendizagem de teste de software," Master's thesis, Universidade Federal de Santa Maria, 2012.

[30] M. Thiry, A. Zoucas, and A. C. da Silva, "Empirical study upon software testing learning with support from educational game." in *International Conference on Software Engineering and Knowledge Engineering*, 2011.

[31] T. P. B. Ribeiro and A. C. R. Paiva, "ilearntest: Jogo educativo para aprendizagem de testes de software," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2014.