PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 07/12

Proposing Innovative Modeling of Testing Related Concepts

Andrew Diniz da Costa
Viviane Torres da Silva
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO – BRASIL

Monografias em Ciência da Computação, No. 07/12 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena May, 2012

Proposing Innovative Modeling of Testing Related Concepts

Andrew Diniz da Costa, Viviane Torres da Silva[1],

Carlos José Pereira de Lucena

[1] Computer Science Department – Federal Fluminense University

acosta@inf.puc-rio.br, viviane.silva@ic.uff.br, lucena@inf.puc-rio.br

Abstract. As the recognition of systematic software testing increases, there is a need to conceive modeling techniques to explicitly document key concerns associated with test cases. Modeling techniques are known as a good way to perform such documentation, because they provide abstractions and a visual notation to represent testing-specific concerns, therefore facilitating the communication among the members of the project team. Based on this context, this research aims to provide a test conceptual framework that represents relevant test concepts which are not considered by well known test approaches, such as TTCN-3 and UML Testing Profile. From this proposal we intend to reduce overall testing time in systems related to different domains. Industrial systems will be used to test and analyze this new approach.

Keywords: Software Testing, Test Modeling, UML Profile.

Resumo. Com o reconhecimento da crescente importância de realizar testes de software, há a necessidade de prover técnicas de modelagem para documentar explicitamente as principais preocupações associadas com testes. As técnicas de modelagem são conhecidas como uma boa maneira de realizar essa documentação, já que fornecem abstrações e uma notação visual para representar específicas preocupações de teste, facilitando assim a comunicação entre membros de equipes. Com base neste contexto, esta pesquisa tem como objetivo fornecer um framework conceitual de teste que represente relevantes conceitos de teste e que não são considerados por conhecidas abordagens de teste, como TTCN-3 e UML Testing Profile. A partir desta proposta pretende-se reduzir o tempo de trabalho na área de teste em sistemas relacionados a diferentes domínios. Sistemas industriais serão utilizados no artigo para testar e analisar esta nova abordagem.

Palavras-chave: Teste de Software, Modelagem de Teste, Perfil UML.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction 1

2 Research Questions and Hypotheses 2

- 2.1 Identifying Relevant Test Concept 2
- 2.2 Modeling Identified Test Conpects 2
- 2.3 Advantages of the Conceptual Framework 3
- 2.4 Generating Scripts of Test 3

3 Research Progress 3

- 3.1 Preliminary Results 3
- 3.2 Current and Next Steps of the Research 5

4 Contributions and Evaluation 6

References 6

1 Introduction

The maintenance and updating of large-scale systems are usually supported by the development of test cases and their documentation. Such documentation is extremely important to track the tests and the changes performed in each system version. Without documenting the test cases it is difficult to plan the necessary investments to perform the tests and also to guarantee that the remaining faults were not caused due to the changes executed while maintaining or updating the software [Black, 2002][Harrold, 2000] [Harrold, 2008]. Therefore, there is a need for techniques that support the documentation of test related concerns [Kaner et al., 1999] [Kaner et al., 2001].

Over the past six years, the Software Engineering Lab of the Pontifical Catholic University of Rio de Janeiro has coordinated and carried out software systems tests developed by different Brazilian companies. From this experience we could identify the importance of documenting specific test concepts. Several approaches have been used in the literature to accomplish this test documentation, such as creating spreadsheets, using test management tools [RQM, 2012][RTM, 2012], and modeling [Feudjio, 2012][OMG, 2012][Trost and Cavarra, 2012][TTCN-3, 2012].

Based on this idea of documenting test concepts from models, a test modeling language should provide abstractions and a visual notation to represent testing-specific concerns that would facilitate the communication among the members of the project team. Although there are several modeling languages that support the modeling of test cases, such as [OMG, 2012][Trost and Cavarra, 2012] and [TTCN-3, 2012], they are not expressive enough to document test concepts considered relevant in test projects [Fink, 2003][Harrold, 2000].

Thus, when the expressiveness of test concepts on test modeling approaches is good, the generation of test scripts from test models becomes interesting. This idea motivated the development of the Model Driven Testing (MDT) [MDT, 2012], which aims to: (i) reduce test maintenance costs, (ii) enhance team communication because the model, test suite and trace provide a clear, unambiguous, and unified view of both the system under test and the test itself and (iii) reduce overall testing time.

However, to successfully apply MDT, test tools should be provided in order to use the advantages offered by such methodology.

In the Software Engineering Lab we often perform tests in self-adaptive systems that apply the self-test concept [b-Costa et al, 2010]. The main idea of the self-test is to execute tests before effectively adapt some behavior. Depending on the system, several test cases are created and maintained. This work involves a high cost which could be reduced by using appropriate tools.

Thus, the initial main focus of this research is to reveal potential gaps or deficiencies of test modeling languages that have been proposed in the literature. Next, we intend to propose a test conceptual framework that represents the deficiencies identified in the analyzed approaches. From this framework, approaches of test modeling will be able to extend it in order to apply relevant test concepts that are not being presently considered.

The paper is organized as follows. Section II presents our proposal in detail. Section III describes the research progress achieved until the current moment. Finally, Section

IV describes the main contributions of this research and how we intend to test some identified hypotheses in it.

2 Research Questions and Hypotheses

Our main goals are to investigate which relevant test concepts are not modeled for the approaches proposed in the literature and how such concepts could be modeled to help on the documentation of test teams. Thus, we intend to answer the following questions.

- Which test concepts are considered relevant in test projects?
- Which of these relevant test concepts have not been considered by well known test modeling approaches proposed in the literature?
- How can we model these concepts?
- Which are the advantages to generating scripts of test cases from the approaches that consider these relevant concepts?

Our research is based on a family of empirical studies to answer these questions and to analyze the four topics discussed in the following subsections. Each topic defines one or more hypothesis that will be evaluated by the research.

2.1 Identifying Relevant Test Concept

We believe that the test modeling languages proposed in the literature do not consider important test concepts in their approaches, revealing potential gaps or deficiencies. Our observation stems from the fact that we have extensively worked on coordinating and carrying out tests of software systems developed for different Brazilian companies in the Software Engineering Lab. From this experience, we could verify that a set of test concepts should be documented and could be represented in models. Based on this idea, our first hypothesis assumes that the test modeling approaches proposed in the literature do not represent a set of relevant test concepts.

Hypothesis, H_1: There is a set of important test concepts proposed in the literature for test modeling languages that have not been considered.

2.2 Modeling Identified Test Conpects

Since several approaches propose the modeling of test concepts, we believe that a test conceptual framework, which represents these relevant test concepts, can be used by such approaches in order to include the not considered concepts. UML Testing Profile (UTP) [OMG, 2012], Agedis [AGEDIS, 2012][Trost, 2012] and UTML [UTML, 2012] are some examples of well known approaches of test modeling. However, we have a special interest in UTP, which has been an OMG official standard for modeling test since 2005 and that has been defined by a consortium of institutions (Ericsson, Fraunhofer/FOKUS, IBM/Rational, Motorola, Telelogic, University of Lübeck). Thus, we intend to evaluate if and how the UTP can be extended to represent expressively the identified test concepts. Besides, we will study which concepts are represented by the UTP and by other approaches in order to test this hypothesis. From this work, we will

propose a test conceptual framework that represents important concepts that have not been considered by known test modeling approaches.

Hypothesis, H_2: UML Testing Profile can be extended to represent expressively new relevant test concepts represented by a new test conceptual framework.

2.3 Advantages of the Conceptual Framework

From the new conceptual framework and the extension of the UTP, we intend to evaluate the advantages of using this new idea of test modeling. We believe that this new approach will offer three advantages in comparison to the other approaches: (i) it will reduce the amount of errors in the modeling; (ii) the effort to represent the new test concepts from the new approach will be lower and (iii) the maintenance work of models created from the new approach will be easier.

To test these hypotheses we intend to perform experiments based on systems where we perform tests; also, we plan to involve people with different testing and modeling skills.

Hypothesis, H_3: The new approach reduces the amount of errors in the modeling compared to using other approaches.

Hypothesis, H_4: The new approach reduces the effort to modeling some test concepts compared to other approaches.

Hypothesis, H_5: The new approach makes the maintenance of models easier than when using other approaches.

2.4 Generating Scripts of Test

From the moment that test modeling languages approaches consider the test concepts represented by our conceptual framework, we believe that they will allow us to generate more complete test scripts from appropriate tools. This idea stems from the fact that the framework will represent relevant test concepts identified from our experience in different test projects, which are not considered by test modeling approaches. To evaluate this hypothesis we intend to study works that perform generation of test scripts. From this evaluation, we will identify which gaps are not treated by such works as well as if our proposal can help on the generation of test scripts.

Hypothesis, H_6: Our approach helps to generate better test scripts than other works provided in the literature.

3 Research Progress

In this section, we further explain the proposed work. To clarify the idea, we detail the preliminary result achieved and the next steps that we intend to follow.

3.1 Preliminary Results

The first stage in our research aimed at investigating whether (or not) there are relevant test concepts that are not supported by modeling languages proposed in the literature and that should be documented. In this context, we conducted an inquisitive study [Lethbridge et al, 2000] described in [a-Costa et al, 2010] that lasted three years.

This study enabled us to identify potential test concepts (see section III.A) that need to be supported by test models and an observational analysis of how such test concepts were important in large-scale software projects. The former was mainly supported by the elaboration of interviews and a questionnaire distributed to several developers with different skills regarding testing and modeling. Then, a list of recurring test concepts identified by developers and testers was defined. This list was used to support a reflective analysis of how existing modeling languages could be enhanced while supporting testing in different software projects.

Figure 1 illustrates the main steps of this study. The first step was about carrying out an initial identification of useful test concepts across several long-term software projects in the domain of petroleum control. Since the initial identification was exploratory, there was no specific constraint on the nature of the testing concepts being gathered. This analytical work was performed during two years, and seven versions were considered in three software projects.

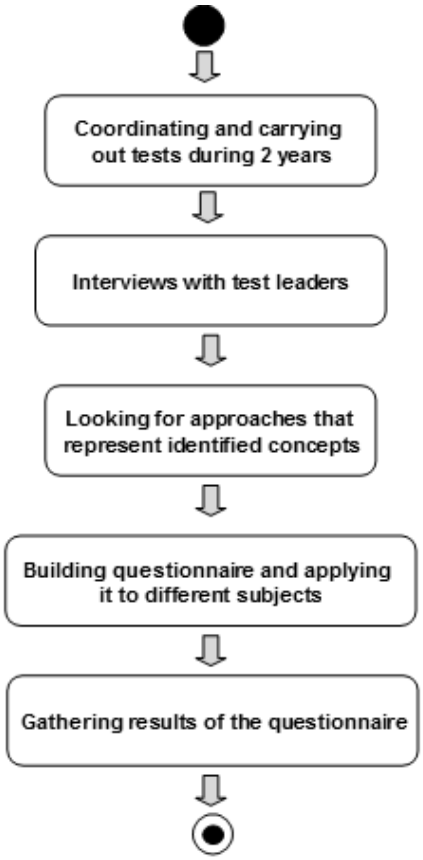

Figure 1. Empirical Procedures that identified neglected test concepts.

In the second step, several interviews with test leaders were conducted. The goal of the interviews was to discuss with them to what extent those initially-elicited concepts were, according to their experience, potentially useful for test modeling and documentation. In the third step we evaluated a set of test modeling languages and management tools proposed in the literature that represented test concepts. The main idea of this study was to identify if such proposals considered the identified test concepts.

The fourth step involved the design of a questionnaire, which followed well-known recommendations [Fink, 2003] and was applied to fourteen invited participants with different testing skills and experience (see Table 1). From this questionnaire, it was

possible to confirm the importance of the test concepts suggested by the test leaders. This questionnaire was validated by two experts on testing modeling and experimental software engineering.

The last and fifth step gathered and analyzed the responses of the applied questionnaire in detail. The results were presented and discussed with the involved stakeholders.

Subjects and their Roles	Academic Background	Years of experience	Description
(5 subjects) 1 project leader 2 senior developers 1 junior developer	2 PhD candidates and 1 MSc in Software Engineering (SE); 1 grad., 1 undergrad. student in Computer Science (CS)	> 3 years	Large knowledge of testing concepts, tools, libraries, e.g. Rational tools, JUnit, DBUnit, so on.
(6 subjects) 2 database admins. 4 senior developers	1 PhD candidate in SE, 3 MSc in SE, 2 grad. in CS	1..3 years	Worked extensively with unit testing and had experience with functional tests and performance tests.
(3 subjects) 2 senior developers 1 junior developer	2 undergrad. students in CS, 1 undergrad. student in SE	< 1 year	Not much experience with tests, but knowledgeable about all test concepts.

Table 1. PARTICIPANTS OF THE INQUISITIVE STUDY.

3.2 Current and Next Steps of the Research

From the inquisitive study described in the previous section we are creating a test conceptual framework that represents these identified test concepts. From this framework, test modeling languages will be able to include the concepts that have not been considered yet. Aiming to exemplify, we have decided to extend UML Testing Profile (UTP). UTP was chosen because we verified that it is the approach that represents more relevant test concepts used in the test projects considered in the inquisitive study described in [a-Costa et al, 2010]. Besides, it is an OMG official standard for modeling test and it was defined by a consortium of institutions, as stated in section II.B.

When the extension of UTP began, we realized that UTP does not allow the modeling of policies which define the access control to modeled test concepts, such as test

cases. This was one of the concepts identified by the inquisitive study mentioned in section III.A.

Accordingly, taking this deficiency into consideration, we evaluated that SecureUML [Basin et al, 2009] could be used to attend this concern. SecureUML is a famous modeling language for formalizing access control requirements that is based on role-based access control. Thus, it provides a language for specifying access control policies for actions on protected resources. However, it leaves open what the protected resources are and which actions they offer to its clients.

These protected resources can be considered test concepts modeled from a test modeling approach. Based on this idea, we decided to use SecureUML with the UTP eXtended (UTX) to represent the control policies to test resources.

Our last step will be to create a tool that makes it possible to generate test scripts from UML diagrams [Booch et al, 2005] created by the UTPX and SecureUML. This tool will be a plug-in of the Rational Software Architect (RSA) [RSA, 2012], tool that enable us to create UML diagrams and that is used in the projects of the Software Engineering Lab.

4 Contributions and Evaluation

The expected contributions of this research are: (1) a list of test concepts that are not currently considered by the modeling languages proposed in the literature, (2) a new test conceptual framework that represents these important test concepts, (3) an extension of the UML Testing Profile from the proposed framework, (4) the use of the SecureUML to model access control policies for actions on protected test resources, (5) evaluating if the inclusion of new concepts in the SecureUML will be necessary to better represent security policies to test resources and (6) the development of a plug-in to the RSA in order to generate test script from diagrams of the UTPX and SecureUML.

As far as evaluation is concerned, controlled experiments involving industrial systems will be performed to analyze more precisely the hypotheses previously defined (sections II.B, II.C and II.D). We plan to model test concepts applied to different systems tested for different test teams. Systems related to different domains and sizes (e.g. amount of requirements developed) will be also used. Besides, we plan to include people with different testing and modeling skills in these experiments in order to evaluate the above mentioned hypotheses.

References

AGEDIS - Automated Generation and Execution of Test Suites for Distributed Component based Software, http://www.agedis.de

Basin, D., Clavel, M., Doser, J., Egea, M. Automated Analysis of Security-Design Models, Journal Information and Software Technology, Volume 51 Issue 5, 2009.

Black, R. 2002: Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, Publisher: Wiley, 2nd edition, ISBN: 0471223980

Booch, G., Rumbaugh, J., and Jacobson, I. Unified Modeling Language User Guide, 2nd Edition, The Addison-Wesley Object Technology Series, 2005.

a-Costa, A. D. Silva, V. T., Garcia, A., Lucena, C. J. P. Improving Test Models for Large Scale Industrial Systems: An Inquisitive Study, Proceedings of the ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems, Part I, LNCS Springer 6394, pp. 301-315, Oslo, Norway, 2010.

b-Costa, A. D., Nunes, C., Silva, V. T., Fonseca, B., Garcia, A., Lucena, C.; JAAF+T: A Framework to Implement Self-Adaptive Agents that Apply Self-Test, Proceedings of the 15th Annual ACM Symposium on Applied Computing 2010, Agent-Oriented Software Engineering Methodologies Infrastructures and Processes (AOMIP) track, volume 2, pp. 928-935, Sierre, Switzerland, 2010.

Feudjio, A. V.: MDTester User Guide, http://www.fokus.fraunhofer.de/distrib/motion/utml/

Fink, A. The Survey Kit: How to ask survey questions, Volume 2, Publisher: SAGE, ISBN 0761925791, 2003.

Harrold, M. J. Testing: A Roadmap. Proceedings of ICSE 2000, Future of Software Engineering, pp. 61-72, 2000.

Harrold, M. J. Testing Evolving Software: Current Practice and Future Promise. Proceedings of ISEC 2008, pp. 3-4, 2008.

Kaner, C., Bach, J., Pettichord, B. Lessons Learned in Software Testing, Publisher: Wiley,1st edition, ISBN: 0471081124, 2001.

Kaner, C., Falk, J., Nguyen, H. Q. Testing Computer Software, Publisher: Wiley, 2nd edition, ISBN: 0471358460, 1999.

Lethbridge, T., Sim, S., Singer, J. Studying Software Engineers: Data Collection Methods for Software Field Studies, Submitted May 2000 to Empirical Software Engineering, 2000.

RSA - Rational Software Architect, http://www.ibm.com/developerworks/rational/products/rsa/

RQM - Rational Quality Manager, http://www-01.ibm.com/software/awdtools/rqm/

RTM - Rational TestManager and Rational ManualTest, http://www-01.ibm.com/software/awdtools/test/manager/

OMG - Object Management Group, UML Testing Profile, version 1, http://www.omg.org/cgi-bin/doc?formal/05-07-07

Pohl, K., Böckle, G., Linder, F. Software Product Line Engineering –Foudations, Principles and Techniques, ISBN-10 3-540-24372-0 Springer Belin Heidelberg New York, 2005.

Trost, J., and Cavarra, A. AGEDIS Language Specification, http://www.agedis.de/documents/d127_1/AGEDIS-ls-fpd.pdf

TTCN-3 web site, http://www.ttcn3.org/

UTML - The Unified Test Modeling Language for Pattern-Oriented Test Design, http://www.fokus.fraunhofer.de/en/motion/ueber_motion/technologien/utml/index.html21.

MDT - Model Driven Testing, https://www.research.ibm.com/haifa/projects/verification/mdt/.