Generation Environment, Execution and Test Management Software Embedded

Marcos Flávio Reis, Ana Maria Ambrosio, Mauricio Ferreira
Instituto Nacional de Pesquisas Espaciais, Brasil
marcosfsreis@gmail.com, ana@dss.inpe.br, mauricio@ccs.inpe.br

Abstract

This paper presents a proposed architecture for creating and managing test embedded system to be used in Software testing lab which is being deployed at INPE. It will be considered as a technique for generating tests, tests based on models, where test cases are generated automatically based on a finite state machine. The architecture also includes the management of test plans, results of their execution, and the management of defects detected. This architecture can contribute to the process of creating and managing tests in the INPE´s testing lab.

1 Introduction

The embedded software development demands a high reliability throughout all its process, especially when used in space projects. These projects have different levels of complexity, increasing the difficulty in validating their functionalities. The improvement of activities and processes of a project of software quality has been the cause of several studies.

One of the approaches proposed for this type of system are tests based on models. A according to Utting [1] is a test strategy where test cases are derived completely or partially from models describing some aspect of a software. One way to accomplish this is through representation of Finite State Machines (FSM). There are some tools which are capable of automatically generating test cases through the representation of a FSM.

Taking into account that the test cases are generated automatically by the FSM, another important step is the test plan, which predicts when, how and who will be involved in testing. The test cases generated automatically also compose the test plan..

It is important that the failures and defects found during the tests are reported at the end of the test process. They will be corrected in the future and will serve as an example.. This effort is treated as Defect Management.

One of the challenges to implement an environment for testing is the integration of all involved activities (Planning, Design, Implementation and Analysis). Another challenge is to maintain and reuse of historic activities that could serve as a knowledge base for future work.

This paper aims to propose architecture to underpin the operation of a test lab of embedded software take into account the variables such as automatic generation of test cases through FSM, planning and design of tests, defect management and knowledge base of tests.

2 Model based testing

Test-Based Model (TBM) consists of a technique for automatically generating a set of test cases with expected inputs and outputs, using models extracted from software requirements [2]. The TBM models the aspects that the system can pass during its operations. One approach is very common for representing this model is by using finite state machine (FSM), which represents the behavior of the system through states, transitions and actions. The FSM can be of two types: Moore and Mealy machine.

3 Test Management

Software test is any activity from the assessment of an attribute or ability of a program or system can be determined whether it achieves the desired results [3].

The test management activity is to control the main steps and testing activities. It basically consists of four activities: Planning, Design / Construction, Implementation and Analysis, as shown in Figure 1. A major product of this management is the test plan, which consists of a set of information that guide or represent the testing process, as requirements, test cases and scenarios [4]. Most of this information is documented in part, scattered in multiple documents and are subject to change as the project evolution

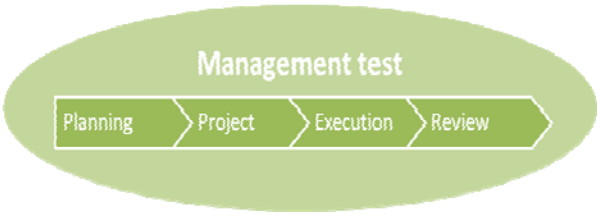

Figure 1: Representation of the phases in a test run

4 Defects Management

Can be defined as a set of processes and procedures that seeks to store and manage information about the bugs and failures found throughout the lifecycle of an application, including from the project until its removal or production output [5].

The main information stored in a defect management is: Identification of the defect, Description, Severity, Priority, Risk associated, Status, and Proof / Evidence of the existence of the defect.

There are free tools for managing defects, but the integration of these tools requires understanding and adapting to the associated process.

5 Architecture

The initial proposal for the architecture to be used in laboratory testing embedded software is a tool for modeling finite state machine capable of generating automatically test cases. For this activity will be used CONDADO tool, created and maintained by Unicamp. Test cases generated will be incorporated into a Open Source test management tool called TestLink [6]. This tool will be created test plans and managing all the tests.

Another important role in the architecture, which is also part of the configuration management application, is the management of defects, faults found in the tests will be reported and managed until the problem is resolved. For this task will be using another OpenSource tool called Mantis [7].

There will be integration between the tools Mantis and TestLink with the aim of linking to the faults found to the test cases executed. This association is important for the knowledge base to be powered by providing information that could be used in future projects as a way to mitigate such problems or proposed solutions. This information knowledge base will be shaped according to the needs encountered during deployment of the architecture.

The Figure 2 represents the architecture described for implementing the laboratory testing of embedded software.

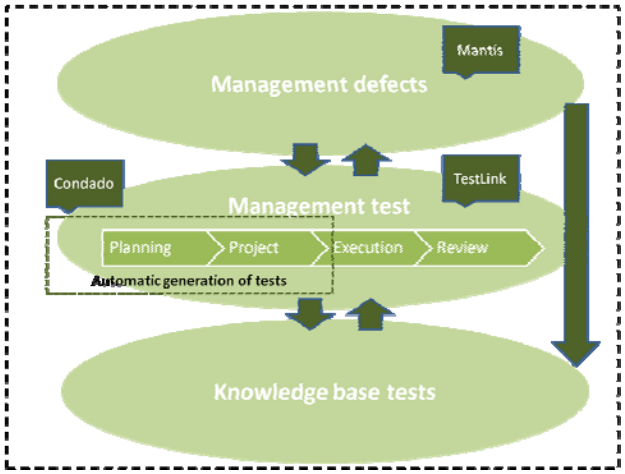

Figure 2: Test Process Architecture

6 Conclusion

The proposed architecture uses key concepts of planning and execution of tests and free tools, interlinked, creating a knowledge base specific to testing embedded software. This environment is presented with an interesting option, both to increase the final quality of software products developed, and to reduce costs, since they are open source tools. Contributions, such as, enrich the laboratory infrastructure in public institutions aiming at verificating and validating an embedded software of high complexity, such as the INPE embedded system..

7 References

[1] Utting and Legeard, Practical model-based testing: a tools approach, Editora Morgan Kauffman, 2007

[2] Binder, R, Testing Object-Oriented Systems: Models, Patterns, and Tools, 2000.

[3] Moreira, Trayahú e Rios, Emerson. Projeto & Engenharia de Software – Teste de Software, Alta Books, 2003

[4] Molinari, Leonardo. Inovação e Automação de Testes de Software, Visual Books, 2010

[5] Molinari, Leonardo. Testes Funcionais de Software, Visual Books, 2008

[6] TestLink available at http://www.teamst.org/

[7] Mantis available at http://www.mantisbt.org/download.php