VVTest: An Environment for Test Information Management to support verification and validation processes

Marcos Flávio S. Reis, Ana Maria Ambrosio, Mauricio Ferreira

National Institute for Space Research [INPE]

São José dos Campos – SP – Brazil

marcosfsreis@gmail.com, ana@dss.inpe.br, mauricio@ccs.inpe.br

Resumo. *This paper describes an environment that integrates tools to generate, execute and manage data of software testing into a single knowledge base. These tools aim to support different activities of verification and validation of CMMi. The proposal uses free tools and unifies the information generated during the planning of tests, the description of the testing requirements and defect management. The environment has been proposed to suit the needs of a critical software systems test laboratory for space applications at INPE.*

1. Introduction

In space projects there are some software which operates on the ground and others on board of satellites. The onboard software is called embedded. In view of the difficulty in developing such software, activities and verification and validation (V&V) are also complex and expensive. Hecht [Hecht, et al., 2005] reported that the cost of V&V activities in space software projects can reach 3% of the total cost of the mission, values considered high for a space program.

As the testing activity consumes much effort of a software project, it is important to increase the efficiency and quality of the tests. [Pressman, 2005] reports that this effort may reach 40% of the total.

Given the large amount of information generated throughout the testing cycle of software, the use of tools to support such activities becomes essential for the smooth progress of work.

This paper presents a solution to support the activities of generating test cases, executing tests and managing the detected defects in a single environment. The solution consists of an environment that integrates existing tools to a unique knowledge base. The tools are: (i) Condado [Martins, ET AL., 1999], which automatically generate test cases from a model of the system in finite state machine; (ii) TestLink [TestLinkCommunity, 2005] that manages the planning and execution of tests, and finally the (iii) Mantis [MantisBT, 2000] for the management of defects found, where they will be accompanied by its detection to correction.

The proposed tool supports processes of verification and validation covered by the Capability Maturity Model Integration (CMMi). This paper presents and compares the

CMMI`s verification and validation processes with the facilities of the proposed environment.

2. Automatic generation of test cases and Condado tool

The model-based testing is a technique that generates software tests from explicit descriptions of behavior of an application [Robinson, 1999]. One way of modeling system using finite state machines (FSM) [Gill, 1962], which serves to specify the behavioral aspect of reactive systems.

The behavior of a system modeled in an FSM is described by states and transitions and can be in only one of its states at any given time [Maldonado, ET AL., 2004]. An extension of the FSM called Extended Finite State Machine (FSME) includes context variables, predictions and actions.

The tool Condado, jointly developed by INPE and State University of Campinas - Brazil (UNICAMP) and recently has received contributions from the Federal University of Lavras - Brazil (UFLA), was developed for the automatic generation of test cases based on FSME. The approach adopted by the tool, with the combination of three types of black box testing - testing state transition, syntax testing and test domain, enables the generation of test covering part of the control and data parameters for transitions [Martins , ET AL., 1999].

The output generated by the Condado is a script containing a sequence of inputs and outputs. Each set of inputs and outputs is a test case consisting of compound paths of transitions between existing states in the EFSM.

A tool for automatic generation of tests from models of states aims to reduce costs and ensure greater coverage of the tests.

3. Test Management and the TestLink tool

A tester needs to know how to plan activities before running the tests. The result of this planning is the test plan, which is a set of information to guide the testing process [Kaner, ET AL. 2001]. An important artifact in the test plan are the test cases that define what will be tested, and serve as a roadmap to be followed by checking one or more test requirements [Santhanam, ET AL., 2002].

The management of the test generates a large amount of information that is utilized more efficiently when handled by a specialist tool.

The tool TestLink created and maintained since 2005 by TestLink Community, available for download at http://www.teamst.org [TestLinkCommunity, 2005], has the resources to manage the tests. In this tool one can manage the plans and test cases, testing requirements, the execution of a battery of tests, allowing to define whether a test was performed correctly or failed, and testers manage platforms.

The TestLink also allows one to associate the defects found during test execution to a defect management tool. There are two other important features in TestLink essential for defining the test environment proposed: the creation of custom fields and import of test case. At first one can define the information necessary for the institution in the testing process. In seconds test cases can be added, and the script generated by the

Condado, are transformed in XML, so the standard set by TestLink can be imported. This allows the integration of the tool for automatic generation of tests and test management.

4. Defect management and Mantis tool

The management of the defects can be defined [Molinari, 2008] as a set of processes and procedures aimed at storing and managing information about the defects found during the life cycle of an application, from design to its withdrawal or exit production.

The definition of error and defect is important for understanding this concept. [Bastos, et al, 2007] defines that an error is the result of human error and defect is the result of an error of a code or a document.

A managed defect means the test manager accompany the defect status since its detection until the resolution of the incident. Besides, it records information, such as identification of the defect, Description, Severity, Priority, Risk associated Status and the evidence of the defect existence.

The Mantis tool, developed in 2000 and maintained by the community MantisBT is a free tool for managing defects and is available on site http://www.mantisbt.org/download.php. It allows creation of custom fields, which are useful for customizing the information according to the environment, and customize the workflow according to the process of the institution.

The Mantis provides users with an interface managers with information for monitoring the media advisory and product defects. The tool also provides the option to export to XML. This feature is used by the environment proposed in this work as a data source for the knowledge base.

This tool allows you to integrate the management of defects to the test management tool provided by TestLink, providing a combination of defects reported in Mantis to test results of TestLink.

5. VVTest

The VVTest allows defining the steps of the management activities of testing and defects in a software testing project. The testing process proposed in [Moreira Filho, ET AL., 2003] basically consists of five phases, as shown in Figure 3: Planning, Preparation, Specification, Execution and Delivery.

The TestLink tool supports all these phases. However, she directs the tester to design or specify test cases. For this reason, the tool is incorporated Condado VVTest to support the automatic generation of test cases. The automation of the specification of test cases through the county not limited to tests that analysts create other types of test cases directly in TestLink, is also possible that other tools can also generate automated test cases that can be entered together.

The defect management has its own activities, open, admission, assignment, resolution and close [Bastos, 2007]. These activities can be seen as steps in a process of defect management. The figure 1 represents the integration stages of the tools used by VVTest.

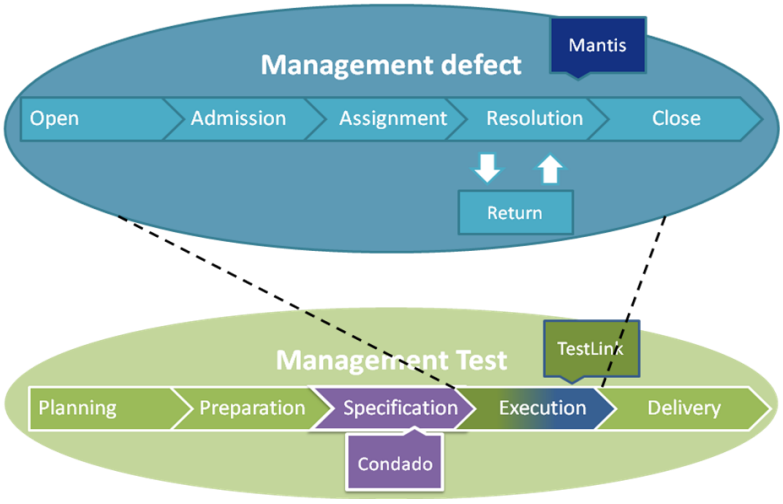

Figure 1 - Integration VVTest

The definition of the activities of the testing process associated with specialist tools are based on the proposed VVTest environment. The VVTest also includes the configuration of each tool according to the needs of information found, the integration of all tools and the creation of a knowledge base consumes all information generated in the tools.

The integration of the tools was made through four modules called: Integration Condado - TestLink Module (MICT), Data Acquisition Module (MAD), Data Insertion Module (MID) and Query Module Data (MCD). These modules were developed taking into account the requirement of interoperability, so that the environment should be ready for future expansion. Figure 2 shows the architecture of integration created by the VVTest.

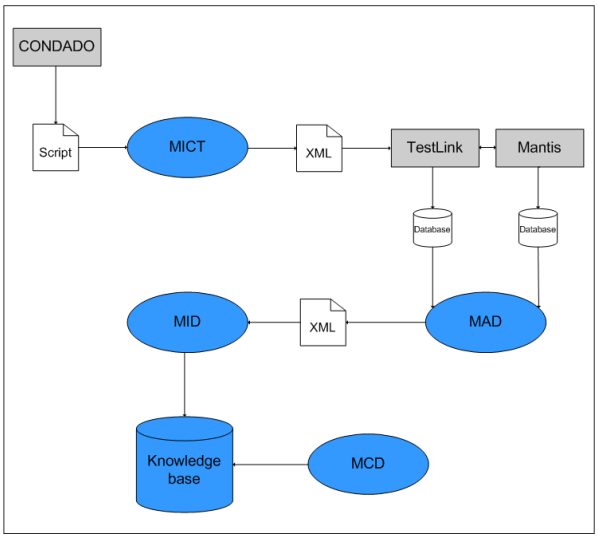

Figure 2- Architecture VVTest

6. Knowledge Base VVTest

Figure 3 presents the conceptual model of the knowledge base used in VVTest and Table 1 contains a description of each entity mentioned in Figure 3.

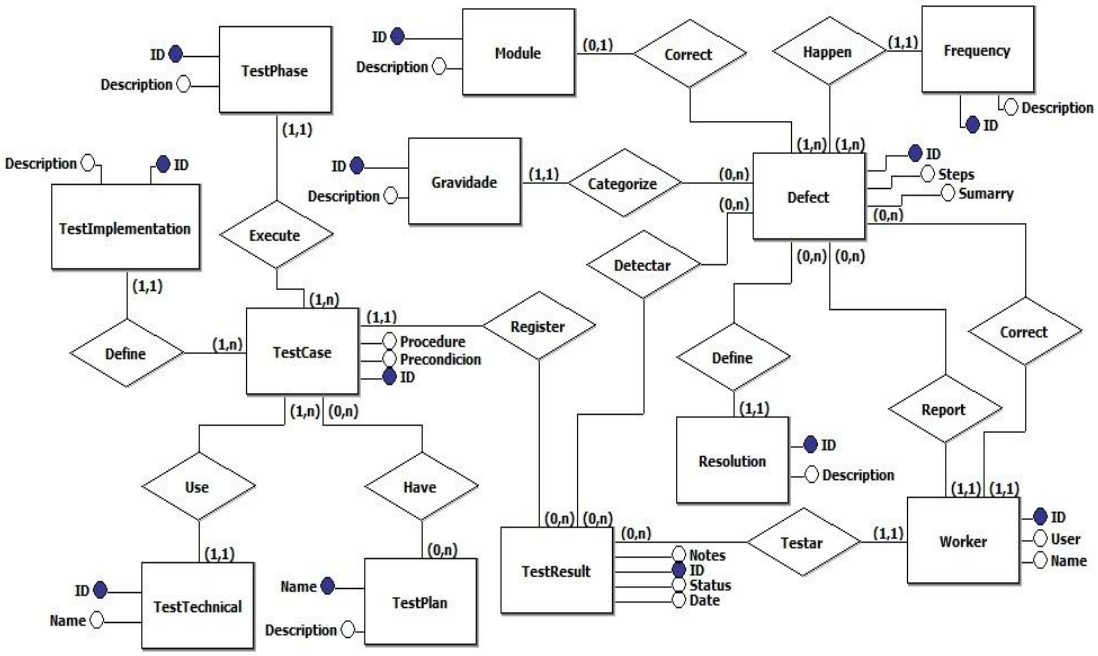

Figure 3 - Conceptual model of knowledge base

Table 1 - Identification of the entities in the knowledge base

Entities	Source of data	Tool	Data example
Test Phase	Test case	TestLink	Design, Architecture
Test Implementation	Test case	TestLink	Manual, Automated
Test Technical	Test case	TestLink	Regression, Load, Functional, Interface
Test Plan	Test plan	TestLink	Name, Summary
Test Case	Test case	Condado and/or TestLink	ID, Summary, Steps
Test Result	Result of running the test case	TestLink	Failed, blocked, passed, and additional information of the result.
Worker	Result of running the test case and Defect	TestLink and Mantis	Name and Login of tester and developer
Defect	Defect	Mantis	ID, Description, Severity, Priority
Module	Defect	Mantis	Class or Module name
Severity	Defect	Mantis	Small, Large, Obstacle
Frequency	Defect	Mantis	Random, Always, sometimes
Resolution	Defect	Mantis	Open, Fixed, duplicate

The knowledge base allows various analyzes, for example, to assess which test technique produced a set of test cases with more power to find fault in the software tested and answer questions such as:

- What test cases were executed at a particular stage of testing?

- What better testing technique to detect high severity defects?
- What is the best technique for detecting errors in a given software module?
- Which tester is more effective with a particular testing technique?

7. Environmental assessment processes to CMMI Verification and Validation

The CMMI [CMMI, 2010] describes that the purpose of Verification (VER) is to ensure that selected work products meet their specified requirements and it defines three goals: Preparing for verification, do peer reviews and verify products selected works.

Table 2 shows how the three practices are met by the planned target SG1. This consists of preparing for the verification, which has as main objective to prepare a framework for the realization of check and set the work products and methods to be used during the process.

The information described in the "What?" In Table 2 can be found in the guide developers of CMMI, version 1.3 [CMMI, 2010].

Table 2 - Evaluation of target SG1 - Prepare for verification

SP 1.1 – Select Work Products for Verification	
What?	Work products are selected based on their contribution to meeting project objectives and requirements, and to addressing project risks. The work products to be verified can include the ones associated with maintenance, training, and support services. The work product requirements for verification are included with the verification methods. The verification methods address the approach to work product verification and the specific approaches that will be used to verify that specific work products meet their requirements **Example Work Products** 1. Lists of work products selected for verification 2. Verification methods for each selected work product
How?	Through VVTest via the TestLink, you can register requirements will be evaluated in verification. If the method of verification is the test and the work products can be modeled by finite state machines, the user can use the Condado to generate the test cases. To assist in decision-making methods adopted in the tests, the test team can consume the information knowledge base, which increases the efficiency of tests through the lessons learned in previous projects.
SP 1.2 – Establish the Verification Environment	
What?	An environment should be established to enable verification to take place. The verification environment can be acquired, developed, reused, modified, or obtained using a combination of these activities, depending on the needs of the project. The type of environment required depends on the work products selected for verification and the verification methods used. A peer review can require little more than a package of materials, reviewers, and a room. A product test can require simulators, emulators, scenario generators, data reduction. **Example Work Products** 1. Verification environment
How?	The VVTest is intended to serve mainly to this practice. Regarding the tests, the environment is formed by the Condado, in the generation of test cases that is automatically integrated with the managements of the tests TestLink and Mantis in the records of defects. The environment can be reused in several projects, and information obtained in previous

	projects can be found both in tools such as the knowledge base. The VVTest is the main focus on support for testing, but the tool TestLink can be used to store information from other methods used in verification, meeting the expectations of this practice.
SP 1.3 – Establish Verification Procedures and Criteria	
What?	Verification criteria are defined to ensure that work products meet their requirements. **Example Work Products** 1. Procedimentos de verificação 2. Critérios de verificação
How?	The procedure can be defined by test case. The tool will provide the Condado of test cases that can be implemented in software. These procedures will be stored in TestLink, which will allow other test case added, increasing the test coverage.

The second goal of this process is the SG2 - Perform Peer Review, whose purpose is that two people together to seek defects in work products and also to detect modifications. For this purpose, this target defines three practices to be performed. Table 3 presents the objectives and how these practices are supported by VVTest .

Table 3 - Evaluation of target SG2 - Make peer reviews

SP 2.1 – Prepare for Peer Reviews	
What?	Preparation activities for peer reviews typically include identifying the staff to be invited to participate in the peer review of each work product; identifying key reviewers who should participate in the peer review; preparing and updating materials to be used during peer reviews, such as checklists and review criteria and scheduling peer reviews. **Example Work Products** 1. Peer review schedule 2. Peer review checklist 3. Entry and exit criteria for work products 4. Criteria for requiring another peer review 5. Peer review training material 6. Selected work products to be reviewed
How?	The VVTest by TestLink can be used to manage the execution of the checklists, and store the entry and exit criteria for each product found. The use of VVTest allows the defects found, properly stored in Mantis, are associated with items of verification. The VVTest via the TestLink also has features (screens and reports) to track the progress of verification, which helps managers in their activities.
SP 2.2 – Conduct Peer Reviews	
What?	Conduct peer reviews of selected work products and identify issues resulting from these reviews. **Example Work Products** 1. Peer review results 2. Peer review issues 3. Peer review data
How?	Problems encountered in peer review may be registered in VVTest by Mantis, allowing the monitoring of this incident until its resolution.
SP 2.3 – Analyze Peer Review Data	
What?	Refer to the Measurement and Analysis process area for more information about obtaining measurement data and analyzing measurement data.. **Example Work Products** 1. Peer review data 2. Peer review action items
How?	The data of defects found can be found at VVTest analytically through each record or summary form, with graphs and metrics generated by the tool.

	The knowledge base stores the information from previous projects, which can aid in defining the action items. The current project can also be found in the knowledge base through the MCD, where you can view other types of graphs.

After checking through a specific method, such as peer review, the standard CMMI defines the third and final goal which is the SG3 - Verify Selected Work Products. In this activity the methods, procedures and criteria are used to verify the product and any associated service. For this, two practices are set for this target, which are supported by VVTest, as shown in Table 4.

Table 4 - Evaluation of target SG3 - Check the Products of Selected Works

SP 3.1 – Perform Verification	
What?	Verifying products and work products incrementally promotes early detection of problems and can result in the early removal of defects. The results of verification save the considerable cost of fault isolation and rework associated with troubleshooting problems. **Example Work Products** 1. Verification results 2. Verification reports 3. Demonstrations 4. As-run procedures log
How?	The VVTest stores the specified requirements, and test procedures, allowing the scanning results are recorded. Have defects found will be reported in Mantis. The VVTest allows you to print reports containing information that is recorded.
SP 3.2 – Analyze Verification Results	
What?	Actual results should be compared to established verification criteria to determine acceptability. The results of the analysis are recorded as evidence that verification was conducted. For each work product, all available verification results are incrementally analyzed to ensure that requirements have been met. Since a peer review is one of several verification methods, peer review data should be included in this analysis activity to ensure that verification results are analyzed sufficiently. Analysis reports or "as-run" method documentation can also indicate that bad verification results are due to method problems, criteria problems, or a verification environment problem. **Example Work Products** 1. Analysis report (e.g., statistics on performance, causal analysis of nonconformances, comparison of the behavior between the real product and models, trends) 2. Trouble reports 3. Change requests for verification methods, criteria, and the environment
How?	Because it is web-based tools, access to information and consequently the results of the verification is simplified in VVTest. Problem reports can be generated. The knowledge base can be used to help define corrective actions, through lessons learned from previous projects, and also in evaluating the results of the current project.

The standard defines three CMMI targets for carrying out the process validation (VAL), which is similar to verification, including using similar methods, for example, the test, the inspection and simulation. What changes between these processes is the vision of the product, the validation is intended to demonstrate that the product meets its intended use when used in its intended environment [CMMI, 2010].

The validation process does not require the practice of peer review, so it has only SG1 - Prepare for validation and SG2 - Validate Product or Product Components. These practices seek the same environment and the same activities of the Verification process, changing only the character of the execution.

Therefore, the proposed environment aids the process of Validation the same way as the Verification process. The integrated environment including the two cases increases the efficiency of records, the preparation of the teams and also provides a more robust knowledge base.

7. Conclusão

The proposed environment uses key concepts for planning and execution of tests and interconnects free tools, creating a knowledge base of tests.

With a centralized data base of knowledge is possible a more thorough analysis of the process of software testing. The comparison of data from the management of defects and testing generates new information. If this information is in a separated way, one would not be able to reach a richer analyses. Moreover, this information is used as a historical basis and can be used to fundament lessons learned for future projects.

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 OpenSource tools. Such contributions enrich the laboratory infrastructure deployed in public institutions that aim to support software verification and validation of high level complexity software. It allows giving the results requested by the reference model CMMi software development, distributed and used worldwide.

References

Bastos, Aderson, ET AL.(2007), Base de conhecimento em teste de software, São Paulo.

Bergeron, B. (2003), Essentials of Knowledge Management. Hoboken.

Gill, A. (1962), Introduction to the Theory of Finite State Machines, New York .

Kaner, Cem, Bach, James and Pettichord, Bret (2001). Lesson Learned in Software Testing, Wiley.

Maldonado, José Carlos, ET AL (2004), Introdução ao teste de Software, São Carlos.

MantisBT (2000). Available at: http://www.mantisbt.org. Accessed May 12, 2012.

Martins, E. , Sabião, S. B. e Ambrosio, A. M. (1999). ConData: A Tool for Automating Specification-Based Test Case Generation for Communication Systems, Software Quality Journal.

Molinari, Leonardo (2008). Testes funcionais de software. Florianopólis.

Moreira Filho, Trayahú e Rios, Emerson (2003), Projeto & Engenharia de Software – Teste de Software, Rio de Janeiro.

Pressman, Roger S. (2005), Software Engineering : A Practitioner Approach, Nova York.

Robinson, Harry (1999), Finite State Model-Based Testing on a Shoestring, Microsoft Corporation.

Santhanam, P e Hailpern, B (2002), Software debugging, testing and verification. IBM System Journal, pp. 4 - 12.

TestLinkCommunity (2005). Available at: http://www.teamst.org/. Accessed May 12, 2012.