# VVTeste: Ambiente de geração e gerenciamento de testes e de defeitos como apoio aos processos de Verificação e Validação do MPS.Br

Marcos F. S. Reis[1], Ana Maria Ambrósio[2], Maurício G. V. Ferreira[2]

[1]Instituto Brasileiro de Tecnologia Avançada (IBTA) – São José dos Campos, SP – Brasil

[2]Instituto Nacional de Pesquisas Espaciais (INPE) – São José dos Campos, SP – Brasil

[3]Departamento de Sistemas e Computação

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

***Abstract.*** *This paper describes an environment for generation, execution and management of software testing data as an aid in fulfilling the objectives outlined by the processes of verification and validation of MPS.Br. The proposed environment uses free tools and defines a single knowledge base in order to unify the information generated during the test planning, test requirements and defect management. This environment will meet the needs of creation and management of software testing.*

***Resumo.*** *Este artigo descreve um ambiente para geração, execução e gerenciamento de dados de testes de software que visa apoiar diferentes atividades dos processos de verificação e validação do MPS.Br. O ambiente proposto utiliza ferramentas livres e utiliza uma base de conhecimento única unificando as informações geradas durante o planejamento dos testes, os requisitos de testes e o gerenciamento de defeitos viabilizando o alcance de novos informações e servindo de fonte de lições aprendidas da organização.*

## 1. Introdução

A utilização de software tornou-se algo comum no dia a dia das pessoas nas mais diversas áreas. Essa aproximação do software às rotinas diárias da comunidade, tem aumentado o nível de qualidade exigido em seu funcionamento, até porque, as falhas ocasionadas por um software podem gerar danos financeiros, de tempo, de reputação das pessoas e empresas e até mesmo a integridade física de seus usuários [ISTQB 2012].

Uma das atividades do ciclo de desenvolvimento de um software que aumenta a qualidade final é a realização de testes. Myers (1979) define testes como um processo de executar um programa com o objetivo de encontrar erro. Entretanto Moreira Filho (2003) descreve que os testes devem verificar se o software está fazendo o que deveria fazer, de acordo com os seus requisitos, e não está fazendo o que não deveria fazer. Essas duas definições são complementares e demonstram que os testes devem garantir o correto funcionamento do sistema, mesmo em condições anormais.

A realização dos testes de software nas organizações é essencial para a finalização do produto, pois revelam defeitos que prejudicarão o funcionamento do software. Estudos, como de o publicado por Myers (1979) ou por Pressman (2005), mostram que quanto antes os defeitos foram encontrados, menor o custo para as organizações. A pior situação é quando um defeito é encontrado em produção, podendo causar grandes prejuízos a um projeto.

A atividade de testes consume boa parte dos recursos de um projeto de software, por isso é importante que seja feito um gerenciamento capaz de aumentar a eficiência e a qualidade dos testes realizados. Pressman (2005) relata que todo o esforço de teste de um software pode chegar a 40% gasto em todo o seu desenvolvimento.

Este artigo apresenta uma solução capaz de apoiar às atividades de geração de casos de teste, execução de testes e gerenciamento de defeitos. O ambiente proposto integra três ferramentas já existentes em uma base de conhecimento única. As ferramentas são:

- Condado [Martins, *et al* 1999] para a geração automática de casos de testes;
- TestLink [TestLinkCommunity 2005] para o gerenciamento do planejamento e da execução dos testes e
- Mantis [MantisBT 2000] para a gestão dos defeitos encontrados.

Além das ferramentas utilizadas para determinadas atividades dos processos de testes, o VVTeste disponibiliza ferramentas para a integração das ferramentas entre si e com uma base de conhecimento de testes, capaz gerar lições aprendidas, novas informações e estatísticas dos projetos de testes realizados na organização.

O ambiente proposto foi desenvolvido buscando as exigências apresentadas pelos processos de Verificação e Validação apresentados pelo Guia de Melhoria do Processo do Software Brasileiro (MPS. BR) [SOFTEX 2011]. Ao final do trabalho é apresentado como o trabalho alcançou os resultados esperados pelo modelo de referência.

## 2. Geração automática de casos de testes e a ferramenta Condado.

O teste baseado em modelos é uma técnica que gera testes de software a partir de descrições explícitas de comportamento de um aplicativo [Robinson 1999]. Uma das formas de modelagem de sistema é através de máquinas de estados finitas (MEF), que serve para especificação do aspecto comportamental de sistemas reativos, particularmente, na área de protocolos de comunicação [Gill 1962].

Um sistema modelado em MEF é descrito por uma máquina composta de estados e transições, estando em somente um de seus estados num dado momento [Maldonado, *et al.*, 2004]. Uma extensão da MEF chamada Máquina de Estado Finitos Estendidos (MEFE) inclui variáveis de contexto, predições e ações.

A ferramenta CONDADO, desenvolvida em conjunto entre INPE e a Universidade de Campinas (UNICAMP) e que recentemente tem recebido colaborações da Universidade Federal de Lavras (UFLA), foi desenvolvida para a geração automática de casos de testes para sistemas de comunicação de protocolo baseado em MEFE. A abordagem adotada pela ferramenta, com a combinação de três testes de caixa preta – teste de transição de estados, teste de sintaxe e teste de domínio, possibilita a geração de teste cobrindo a parte do controle e os dados dos parâmetros de interações do protocolo [Martins 1999]. Uma ferramenta de geração automática de testes a partir de modelos de estados visa diminuir os custos e garantir uma maior cobertura dos testes.

A saída gerada pela CONDADO é um script contendo uma sequência de entradas com suas respectivas saídas esperadas. Cada conjunto de entradas e saídas constitui um caso de teste formado por caminhos compostos de transições entre os estados existentes na MEFE. O padrão empregado define os comandos de entrada como *input* e os de saída como *output.*

## 3. Gerenciamento dos testes e a ferramenta TestLink

Um testador precisa saber planejar as atividades antes da execução dos testes. O fruto desse planejamento é o plano de testes, que é um conjunto de informações que guiam o processo de testes [Kaner 2001]. Um artefato importante no plano de testes são os casos de testes, que definem o que será testado, e servem como um roteiro a ser seguido, verificando um ou mais requisitos de testes [Santhanam 2002].

O gerenciamento dos testes gera uma grande quantidade de informações, que são aproveitadas de maneira mais eficiente quando manipuladas através de uma ferramenta especialista.

A ferramenta TestLink criada e mantida desde 2005 pela TestLink Community, disponível para download em http://www.teamst.org, dispõe dos recursos necessários para o gerenciamento dos testes. Nesta ferramenta é possível gerenciar os planos e casos de testes, os requisitos de testes, a execução de uma bateria de testes, permitindo definir se um determinado teste foi executado corretamente ou com falha além de gerenciar plataformas de testes e testadores.

A TestLink possui três funcionalidades essenciais para as necessidades apresentadas pelo ambiente VVTeste. A primeira é a capacidade de integração com uma ferramenta de gerenciamento de defeitos, permitindo que durante a execução dos testes sejam associados.

A TestLink também permite associar os defeitos encontrados durante a execução dos testes a uma ferramenta de gerenciamento de defeitos. Existem dois outros recursos importantes na TestLink essenciais para definição do ambiente de teste proposto: a criação de campos personalizados e a importação de caso de testes. No primeiro é possível definir as informações necessárias para a instituição no processo de teste. No segundo é possível incluir os casos de teste propriamente dito. Desta forma, o script gerado pela CONDADO, se transformado em XML no padrão definido pela TestLink, pode ser importado. Isso permite a integração da ferramenta de geração automática de testes e a de gerenciamento de testes.

## 4. Gerenciamento de defeitos e a ferramenta Mantis

O gerenciamento dos defeitos pode ser definido [Molinari 2008] como um conjunto de processos e procedimentos que visa armazenar e gerenciar a informação a respeito dos defeitos encontrados ao longo do ciclo de vida de uma aplicação, desde o projeto até a sua retirada ou saída de produção.

A definição de erro e defeito é importante para a compreensão deste conceito. Segundo Bastos (2007), um erro é o resultado de uma falha humana e defeito é o resultado de um erro existente num código ou num documento.

Um defeito quando gerenciado, é acompanhado desde a sua detecção até a resolução deste incidente e registra várias informações, tais como: Identificação do defeito, Descrição, Severidade, Prioridade, Risco associado, Status, Provas e evidências da existência do defeito.

A ferramenta Mantis, desenvolvida em 2000 e mantida pela comunidade MantisBT, é uma ferramenta livre para o gerenciamento de defeitos e está disponível no site http://www.mantisbt.org/download.php. Ela permite a criação de campos personalizados, que são úteis para personalização das informações de acordo com o ambiente, e personalizar o fluxo de trabalho de acordo com o processo da instituição.

A Mantis disponibiliza aos usuários gerentes uma interface com informações do acompanhamento dos defeitos e *realeses* do produto, além de permitir a integração do gerenciamento de defeitos ao gerenciamento de teste provido pela ferramenta TestLink, disponibilizando uma associação dos defeitos relatados na Mantis aos resultados de testes da TestLink.

## 5. Arquitetura do ambiente VVTeste

As ferramentas TestLink e Mantis já possuem facilidades para integração nativa. Com algumas alterações em seus arquivos de configuração as duas ferramentas são integradas, permitindo que durante a execução dos testes, o testador faça a associação com um ou vários defeitos que estão registrados na Mantis. Já a integração da ferramenta CONDADO com as outras ferramentas exigiu algum esforço.

O ambiente VVTeste foi concebido em 4 módulos: (i) Módulo Integração Condado x TestLink (MICT), (ii) Módulo Inclusão de dados (MID), (iii) Módulo Aquisição de dados (MAD) e (iv) Módulo Consulta de dados (MCD). Além dos módulos, o ambiente tem uma base de conhecimento única para armazenar todas as informações adquiridas durante os projetos de testes. A Figura 1 apresenta a arquitetura do ambiente VVTeste.

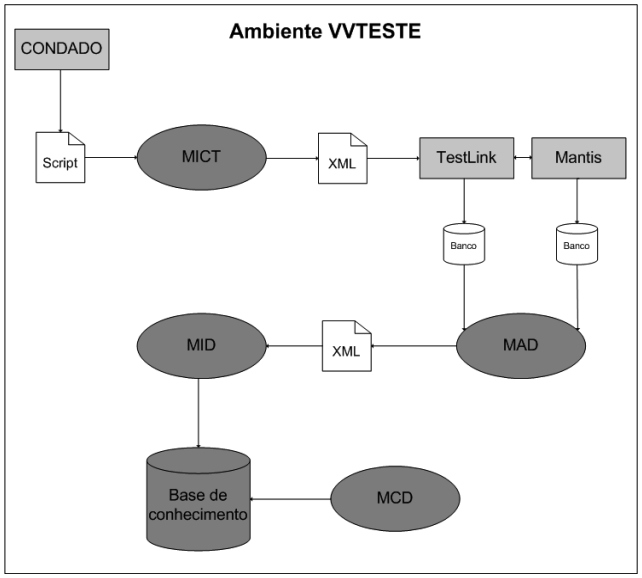


**Figura 1 - Ambiente VVTeste**

As caixas em cinzas representam as ferramentas existentes utilizadas pelo ambiente. As figuras em branco representam os arquivos ou as fontes de informações que são utilizadas para a integração das ferramentas e os módulos criados para a integração das ferramentas no ambiente VVTeste estão representados por círculos azuis.

O módulo MICT é responsável pela transformação do script de teste gerado pela Condado em um XML, que posteriormente é importado na TestLink.

O MAD é o responsável por obter as informações nos bancos de dados das ferramentas Mantis e TestLink e gerar um conjunto de arquivos XMLs com todas essas informações. Já o MID faz o processamento dos XMLs gerados pelo MAD e inclui as informações obtidas na Base de conhecimento. O MAD e o MID foram desenvolvidos separadamente para permitir interoperabilidade ao ambiente, pois assim outras ferramentas também podem adicionar informações a base de conhecimento.

A base de conhecimento unifica as informações geradas durante a gerência dos testes e o controle dos defeitos, além de permitir que essas informações sejam cruzadas e consumidas de uma forma que não seria possível com as ferramentas isoladas. Já o MCD é uma ferramenta capaz de executar comandos SQL sobre a base de conhecimento e transformar o resultado em gráficos de colunas ou de pizza [Reis 2012].

A Figura 2 apresenta uma parte da modelo conceitual da base de conhecimento que recebe as informações para serem processadas.

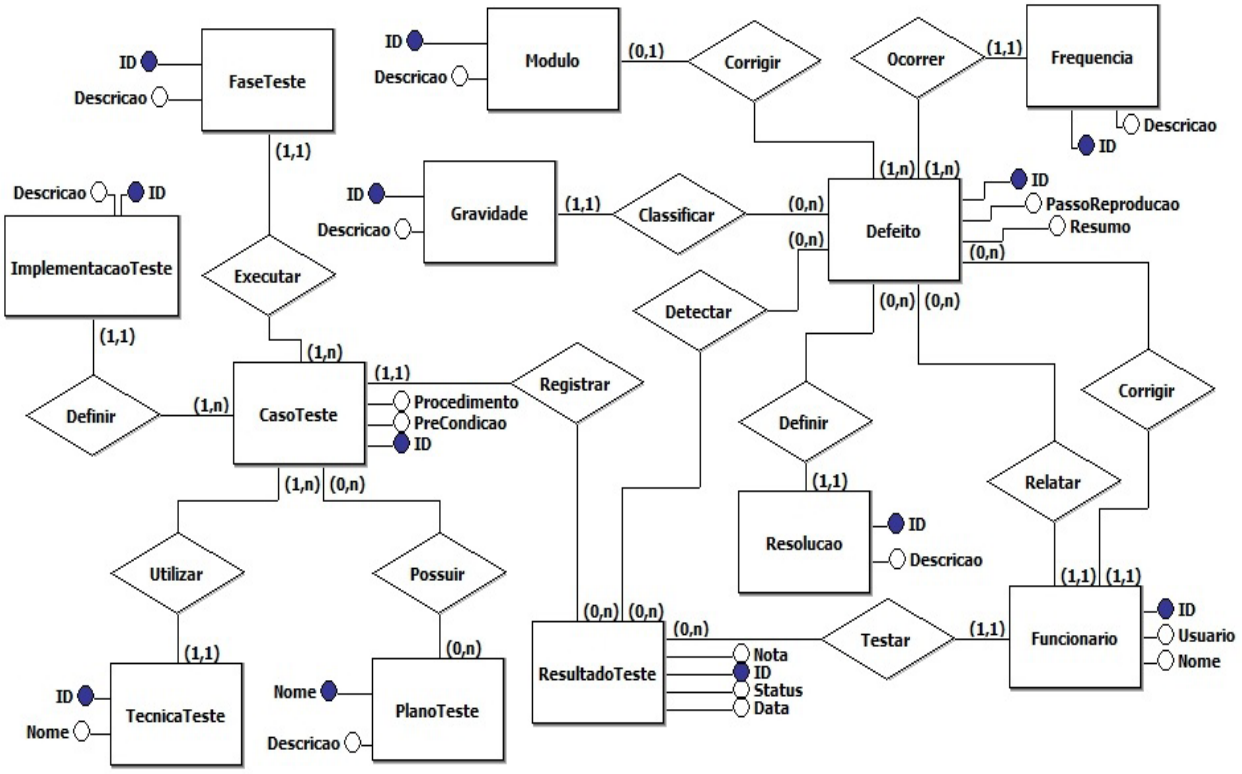


**Figura 2 - Modelo conceitual da base de conhecimento**

## 6. Avaliação do ambiente aos processos de Verificação e Validação do MPS.Br

A Melhoria do Processo de Software Brasileiro (MPS.Br) é um esforço realizado desde de 2003 por uma comunidade liderada pela SOFTEX para desenvolver as micros, pequenas e médias empresas de desenvolvimento de software brasileira. Neste guia, a MPS.Br descreve quais são os resultados esperados para os processos de Verificação e Validação de Software.

A Verificação tem o propósito de confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto atende apropriadamente os requisitos especificados [MPS.Br 2011]. Para tanto, são definidos seis resultados esperados (RE) para este processo.

O propósito do processo Validação é confirmar que um produto ou componente do produto atenderá a seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido [MPS.Br 2011]. Para atender este processo são definidos sete REs.

Os resultados esperados pelo MPS.Br para estes dois processos são parecidos, apenas sendo adaptados ao foco da atividade. A avaliação feita, de acordo com o ambiente VVTeste criado, é apresentado na Tabela 1.

**Tabela 1 - Avaliação do VVTeste aos processos do MPS.BR [Reis 2012]**

| **VER1/VAL1** |
|---|
| Os requisitos são consultados ou alterados na ferramenta TestLink. Outro fator importante, é que informações de projetos anteriores podem ser adquiridas tanto na TestLink como na Base de conhecimento para ajudar a avaliar a complexidade dos produtos de trabalho. |
| **VER2/VAL2** |
| As ferramentas auxiliam em algumas atividades desses processos. Além disso, o ambiente foca apenas na técnica de teste. O objetivo é gerar subsídios para a preparação da estratégia de V&V. Para a geração dos casos de testes o ambiente disponibiliza a ferramenta Condado para os produtos de trabalho que possam ser modelados através de MEFEs. Neste caso, eles são criados automaticamente e integrados à ferramenta de gerenciamento dos testes, a TestLink.<br>O VVTeste, via a TestLink tem um papel muito importante nessa atividade, porque além de consumir os casos de testes gerados pela Condado, ele permite que outros casos sejam descritos. Além disso, ele também permite a divisão do trabalho entre os envolvidos na avaliação.<br>Todas as informações referentes aos métodos adotados e informações da estratégia adotada que esteja diretamente ligada à execução da avaliação são descritas na Testlink, a qual pode ser facilmente consultada por todos os membros envolvidos. |
| **VER3/VAL 3** |
| Os critérios e procedimentos podem ser armazenados no VVTeste, via a TestLink, juntamente com cada um dos casos de testes gerados.<br>Conforme descrito pelo modelo, os aspectos importantes para o ambiente estabelecido é que ele seja capaz de gerenciar o planejamento e a execução dos testes. Isso também pode ser feito através da TestLink.<br>Este ambiente não contempla nenhuma forma automatizada de execução dos testes, pois normalmente este tipo de ferramenta depende muito da arquitetura utilizada no desenvolvimento do software, sendo assim, caso a organização deseje automatizar a execução, deverá avaliar uma ferramenta que atenda as características do produto. |

| VER4/VAL4 |
| --- |
| Durante a execução das atividades as informações são consultadas e dirigidas no VVTeste, através da TestLink. Nela os executores registram quais os casos de testes foram executados com sucesso, quais falharam e os que foram impedidos de serem avaliados. O gerente acompanha, através dos gráficos e das interfaces da TestLink, o andamento das atividades e compara o andamento cronograma estipulado. |
| **VER5/VAL5** |
| O VVTeste, via a Mantis, registre esses defeitos. A integração entre TestLink e Mantis, permite relacionar os casos de testes aos defeitos encontrados. A Mantis permite o acompanhamento do defeito/problema até a sua resolução, aumentando a segurança sobre os relatos registrados. |
| **VER6/VAL6** |
| As ferramentas Mantis e TestLink disponibilizam diversos relatórios e gráficos que auxiliam na análise dos resultados.<br>Além disso, através da Base de conhecimento e do módulo MCD, as resultados podem ser analisados de outras formas, inclusive comparando com projeto ou baterias de testes anteriores. |
| **VAL7** |
| Uma das formas de evidenciar os produtos é apresentar um relatório dos problemas encontrados e solucionados. Este relatório pode ser adquirido através da Mantis, que permite a exportação de alguns dados para outros formatos, como por exemplo, o Word, e disponibiliza alguns dos gráficos produzidos. |

## 7. Conclusão

O ambiente proposto utiliza os principais conceitos de planejamento e execução de testes e ferramentas livres interligadas, gerando uma base de conhecimento de testes.

Com os dados centralizados na base de conhecimento é possível uma análise mais apurada do processo de testes de software. O cruzamento dos dados das gerências de defeitos e de testes gera novas informações, que separadas não seriam capazes de serem alcançadas. Além disso, essas informações são utilizadas como base histórica e lições aprendidas para projetos futuros.

Este ambiente se apresenta com uma opção interessante, tanto para aumentar a qualidade final dos produtos de software desenvolvidos, quanto para reduzir custos, por se tratarem de ferramentas *OpenSource*. Além disso, apesar de não atender todas as necessidades dos processos de Verificação e Validação, o ambiente proporciona ferramentas para apoiar tais processos, principalmente quando a técnica utilizada é o teste.

## Referências

Albuquerque, Jade e Moreira, Albert (2006), "Gestão do conhecimento e tecnologia da informação: As contribuições dos sistemas especialistas".

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.

ISTQB, "Base de conhecimento de Testes de Software" (2012). http://www.bstqb.org.br/. Acessado em: 26 de agosto de 2012.

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). http://www.mantisbt.org. Acessado em 27 de setembro de 2012.

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

Myers, Glenford. (1979), "The art of software testing", Nova York.

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

Reis, M. F. S. "VVTESTE: Ambiente de geração e gerenciamento de testes e de defeitos de software como apoio a processos de verificação e validação". http://urlib.net/8JMKD3MGP7W/3BESFE5. Acesso em: 26 agosto de 2012.

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

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

SOFTEX, "Guia de Implementação – Parte 4: Fundamentação para Implementação do Nível D do MR-MPS", http://www.softex.br/mpsbr/_guias/guias/ MPS.BR_Guia_de_Implementacao_Parte_4_2011.pdf . Acessado em 27 de setembro de 2012.

TestLinkCommunity (2005). http://www.teamst.org/. Acessado em 27 de setembro de 2012.