UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AVALIAÇÃO DO SUPORTE FORNECIDO POR FERRAMENTAS DE GESTÃO DE TESTE AO PROCESSO DE TESTE

TRABALHO DE GRADUAÇÃO

Gabriel Kist Brusius

Santa Maria, RS, Brasil

2011

AVALIAÇÃO DO SUPORTE FORNECIDO POR FERRAMENTAS DE GESTÃO DE TESTE AO PROCESSO DE TESTE

por

Gabriel Kist Brusius

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Profª. Lisandra Manzoni Fontoura

Trabalho de Graduação N. 320
Santa Maria, RS, Brasil
2011

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

AVALIAÇÃO DO SUPORTE FORNECIDO POR FERRAMENTAS DE GESTÃO DE TESTE AO PROCESSO DE TESTE

Elaborado por
Gabriel Kist Brusius

como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Profª. Lisandra Manzoni Fontoura
(Presidente/Orientador)

Profª. Oni Reasilvia de Almeida Oliveira Sichonany (UFSM)

Profª. Márcia Pasin (UFSM)

Santa Maria, 14 de julho de 2011

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelas oportunidades que me foram dadas na vida, principalmente por ter perto pessoas maravilhosas, mas também por ter vivido fases difíceis, que foram matérias-primas de aprendizado.

É difícil agradecer todas as pessoas que de algum modo nos momentos serenos e/ou apreensivos, fizeram parte da minha vida, por isso agradeço a todos de coração.

Agradeço aos meus pais que me deram a vida e me ensinaram a vivê-la com dignidade. As minhas irmãs, pelo incentivo e apoio para que tudo desse certo.

Aos meus amigos que mesmo longe, estavam perto, dispostos a ajudar e tornar os momentos mais divertidos. Agradeço aos meus colegas de curso, tanto da UNIPAMPA quanto da UFSM pelas horas de companheirismo e por dividir dúvidas, aprendizados e distrações.

Agradeço a professora Lisandra pela orientação, disponibilidade e conhecimento para realização desse trabalho. As professoras Márcia e Oni pelas críticas e sugestões para a realização dessa pesquisa.

Em especial agradeço a minha namorada Angela, por ter vivenciado comigo passo a passo todos os detalhes deste trabalho, ter me ajudado durante o decorrer dessa pesquisa, por ter me dado todo o apoio que necessitava nos momentos difíceis, todo carinho, respeito, por ter me aturado nos momentos de estresse, e por tornar minha vida cada dia mais feliz.

“A adversidade desperta em nós capacidades que, em circunstâncias favoráveis, teriam ficado adormecidas.”

- Quinto Horácio Flaco

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

AVALIAÇÃO DO SUPORTE FORNECIDO POR FERRAMENTAS DE GESTÃO DE TESTE AO PROCESSO DE TESTE

Autor: Gabriel Kist Brusius
Orientador: Profª Lisandra Manzoni Fontoura
Local e data de defesa: Santa Maria, 14 de julho de 2011.

Teste de *software* é definido como um processo de executar um programa com o objetivo de revelar a presença de erros. Visto que, o desenvolvimento de *software* é uma atividade complexa, ferramentas de teste podem ser usadas para suportar as atividades relacionadas à validação de *software*, desde a definição de casos de teste, no início do desenvolvimento, até a etapa de teste, que ocorre no final do desenvolvimento, visando garantir que o *software* desenvolvido satisfaça os requisitos definidos pelo cliente e não apresente defeitos. Existem diferentes ferramentas disponíveis para auxiliar a atividade de teste, dificultando a escolha da ferramenta mais adequada às necessidades do projeto. Visando facilitar a análise de ferramentas para gestão de teste, esse trabalho define, com base na literatura, um conjunto de critérios para avaliação de ferramentas de gestão de teste. Assim, o objetivo deste trabalho é identificar esse conjunto de critérios que possam ser usados para avaliar ferramentas de gestão de teste e avaliar algumas ferramentas analisando os resultados obtidos. No presente estudo, foram escolhidas três ferramentas de gestão de teste, para a aplicação dos critérios de avaliação das mesmas, sendo duas dessas *open source* e uma proprietária. Na análise dos resultados, pode-se perceber uma maior eficácia no suporte das atividades necessárias para o processo de teste em uma ferramenta *open source*, apesar da ferramenta proprietária apresentar inúmeros recursos de aperfeiçoamento, gerando funcionalidades mais completas, como por exemplo, na atividade de emissão de relatórios.

Palavras-chave: Teste de *software*, ferramentas de teste, critérios de avaliação.

ABSTRACT

Graduation Work
Graduation Program in Computer Science
Federal University of Santa Maria

EVALUATION OF THE SUPPORT PROVIDED BY THE TEST MANAGEMENT TOOLS TO THE TESTING PROCESS

Author: Gabriel Kist Brusius
Advisor: Professor Lisandra Manzoni Fontoura

Software testing is defined as the process of executing a program with the aim of revealing the presence of errors. Since software development is a complex task, test tools can be used to support activities related to validation of software, from the definition of test cases, at the beginning of development, until test step, which occurs in end of development, aiming to ensure that the software developed meets the requirements set by the client and it does not features defects. There are different tools available to assist the testing activity, making it difficult choosing the most appropriate tool to project needs. To facilitate the analysis of test management tools, this paper defines, based on the literature, a set of criteria for evaluation of test management tools. Therefore, the objective of this work is to identify this set of criteria that can be used to evaluate management tools test and evaluate some tools analyzing the results. In this work, we selected three test management tools, for the application of the criteria for their evaluation, two of these being open source and one proprietary. In analyzing the results, one can realize greater effectiveness in supporting the activities required for the testing process in an open source tool, although the proprietary tool features include a number of improvements generating more complete functionality, such as the activity reporting.

Keywords: Software testing, testing tools, evaluation criteria.

LISTA DE FIGURAS

Figura 1 Defeito x Erro x Falha 16
Figura 2 Modelo 'V' descrevendo o paralelismo entre as atividades de desenvolvimento e teste de *software* 18
Figura 3 Tela Principal do Testlink 25
Figura 4 Integração Bugzilla e Testopia 26
Figura 5 Tela Principal do Testopia 28
Figura 6 Tela Principal do Rational TestManager 30

LISTA DE ABREVIATURAS E SIGLAS

CSTE	*Certified Software Tester*
CSV	*Comma-separated Values*
HTML	*HyperText Markup Language*
IBM	*International Business Machines*
ID	*Identity Document*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
ISO	*International Organization for Standardization*
PDF	*Portable Document Format*
PHP	*Hypertext PreProcessor*
RUP	*Rational Unified Process*
UFSM	Universidade Federal de Santa Maria
XML	*Extensible Markup Language*

LISTA DE APÊNDICES

Apêndice A – Casos de Testes desenvolvidos 53

SUMÁRIO

1 Introdução 13

1.1 Objetivo 14

1.2 Organização do texto 14

2 Revisão Bibliográfica 15

2.1 Testes de *Software* 15

2.1.1 Terminologia e Conceitos Básicos 15

2.1.2 Níveis de Teste de *Software* 16

2.2 Ferramentas de Teste de *Software* 18

2.3 Ferramentas Escolhidas 21

2.3.1 *Testlink* 21

2.3.2 *Testopia* 25

2.3.3 *Rational TestManager* 28

2.4 Critérios de Avaliação de Ferramentas de Teste 31

3 Materiais e Métodos 32

4 Resultados 35

4.1 Projetando Casos de Teste 35

4.1.1 Ferramenta *Testlink* 36

4.1.2 Ferramenta *Testopia* 37

4.1.3 Ferramenta *TestManager* 38

4.2 Construindo Casos de Teste 38

4.2.1 Ferramenta *Testlink* 40

4.2.2 Ferramenta *Testopia* 40

4.2.3 Ferramenta *TestManager* 40

4.3 Executando Casos de Teste 41

4.3.1 Ferramenta *Testlink* 42

4.3.2 Ferramenta *Testopia* 42

4.3.3 Ferramenta *TestManager* 43
4.4 Acompanhamento dos Resultados da Execução dos Testes 43
4.4.1 Ferramenta *Testlink* 44
4.4.2 Ferramenta *Testopia* 44
4.4.3 Ferramenta *TestManager* 44
4.5 Relatórios de Resultados de Teste 45
4.5.1 Ferramenta *Testlink* 46
4.5.2 Ferramenta *Testopia* 46
4.5.3 Ferramenta *TestManager* 47
4.6 Análise Geral 47
5 Considerações Finais 49
6 Referências 51

1 INTRODUÇÃO

No âmbito computacional, o teste de *software* é definido como um processo de executar um programa com o objetivo de revelar a presença de erros, contribuindo para aumentar a confiança de que o *software* desempenha funções especificadas e representando a última revisão de especificação, projeto e codificação (PRESSMAN, 2006).

De forma geral, o teste de *software* pode ser visto como uma das atividades do processo de qualidade de *software*. A qualidade da aplicação pode, e normalmente varia significativamente de sistema para sistema. No entanto, mensurar o bom funcionamento de um *software* envolve compará-lo com elementos como especificações, outros *softwares* da mesma linha, versões anteriores do mesmo produto, inferências pessoais, normas relevantes, leis aplicáveis, entre outros (MYERS, 2004).

Dentre as implicações referentes ao teste, têm-se os casos de teste, que representam um conjunto de condições usadas para teste de *software*. Ele pode ser elaborado para identificar defeitos na estrutura interna do *software*, através de situações que exercitem adequadamente todas as estruturas utilizadas na codificação; ou ainda, garantir que os requisitos do *software* que foi construído sejam plenamente atendidos (KOSCIANSKI, 2007).

Visto que, o desenvolvimento de *software* é uma atividade complexa, é desejável o uso de ferramentas de teste, que servem para suportar atividades relacionadas ao ciclo de vida de desenvolvimento de *software*: da concepção à implantação (CAETANO, 2007). Elas ajudam as equipes de desenvolvimento a investigarem erros no *software*, verificarem suas funcionalidades e se certificarem a respeito da confiabilidade e da segurança do *software* desenvolvido. As ferramentas estão disponíveis para todos os estágios de um projeto de desenvolvimento de *software* (CENTERS, 2009).

Muitas vezes é difícil escolher a ferramenta para gestão de teste mais adequada às necessidades do projeto e às características da equipe de desenvolvimento. Visando facilitar essa tarefa este trabalho propõe um conjunto de critérios para avaliação de ferramentas de gestão de teste elaborado a partir da literatura. Para a aplicação efetiva de um critério de teste, é necessária a existência de uma ferramenta de apoio. Através do uso de ferramentas, é possível obter uma maior eficácia e uma redução do esforço necessário para a realização do teste, bem como diminuir o número de erros causados pela condução nessa atividade (BARBOSA, 2001).

Devido à diversidade de critérios de teste existentes e a necessidade desses critérios serem aplicadas em conjunto, surge a questão de qual estratégia de teste utilizar, ou seja, como escolher os critérios de teste, de forma que as vantagens de cada um desses critérios sejam combinadas para se obter o melhor resultado (GHOSH, 2001).

1.1 Objetivo

O objetivo deste trabalho é identificar um conjunto de critérios para avaliação de ferramentas de gestão de teste, e analisar algumas ferramentas com base nesses critérios, comparando os resultados obtidos.

A análise de resultados da comparação visa facilitar a escolha da ferramenta de gestão de projetos por parte da equipe de desenvolvimento.

1.2 Organização do texto

O texto está organizado como segue. No capítulo 2, são apresentados tópicos básicos para o entendimento do trabalho sobre testes de *software*, incluindo terminologias e conceitos, ferramentas de teste e critérios de avaliação. No capítulo 3, são apresentados os materiais e métodos utilizados para o desenvolvimento do presente estudo. No capítulo 4, são expostos os resultados obtidos na aplicação dos objetivos da pesquisa, bem como uma comparação entre as ferramentas analisadas. O capítulo 5 descreve as considerações finais. Ao final do texto, encontram-se os apêndices referenciados no decorrer do texto.

2 REVISÃO BIBLIOGRÁFICA

Nesse capítulo estão descritos conceitos teóricos que fundamentam este trabalho.

2.1 Testes de *Software*

Teste de *software* é o processo de execução de um produto de *software* para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pelas equipes de desenvolvimento antes da entrega final (NETO, 2008).

Testar um *software* significa verificar através de uma execução controlada se o seu comportamento ocorre de acordo com o especificado. O objetivo principal é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos (NETO, 2008).

2.1.1 Terminologia e Conceitos Básicos

As definições que serão usadas a seguir seguem a terminologia padrão para Engenharia de Software do IEEE – *Institute of Electrical and Electronics Engineers* (IEEE, 1990). Na Figura 1 pode-se verificar a relação entre defeito, erro e falha, que são definidos como:

- Defeito (*fault*): passo, processo ou definição de dados incorretos, como uma instrução ou comando incorreto.
- Erro (*error*): diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro.
- Falha (*failure*): é o comportamento operacional do *software* diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

Figura 1 Defeito x Erro x Falha
Fonte: (NETO, 2008).

A atividade de teste é composta por elementos essenciais que auxiliam na formalização desta atividade. Esses elementos são os seguintes:

- Requisitos: expressam as características e restrições do produto de *software* do ponto de vista de satisfação do usuário (MACORATTI, 2006).
- Plano de Teste: é um documento que descreve os objetivos, escopo, abordagem e foco de um esforço de teste de *software* (SOFTWAREQATEST, 2011).
- Caso de Teste: descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado (CRAIG, 2002).
- Procedimento de Teste: é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste (CRAIG, 2002).
- Critério de Teste: serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto (ROCHA, 2001).

2.1.2 Níveis de Teste de *Software*

O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do *software*. Sendo assim, os principais níveis de teste de *software* são (ROCHA, 2001):

- Teste de Unidade: também conhecido como testes unitários. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.
- Teste de Integração: visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do *software* que foi estabelecida na fase de projeto.
- Teste de Sistema: avalia o *software* em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do *software*. Verifica se o produto satisfaz seus requisitos.
- Teste de Aceitação: são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.
- Teste de Regressão: não corresponde a um nível de teste, mas é uma estratégia importante para redução de "efeitos colaterais". Consiste em se aplicar, a cada nova versão do *software* ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de testes anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

O planejamento de projeto dos testes deve ocorrer seguindo os seguintes passos:

a) Inicialmente é planejado o teste de aceitação a partir do documento de requisitos;

b) Após isso, é planejado o teste de sistema a partir do projeto de alto nível do *software*;

c) Em seguida, ocorre o planejamento dos testes de integração a partir do projeto detalhado.

d) E, por fim, o planejamento dos testes a partir da codificação.

Esta estratégia é sumarizada na Figura 2.

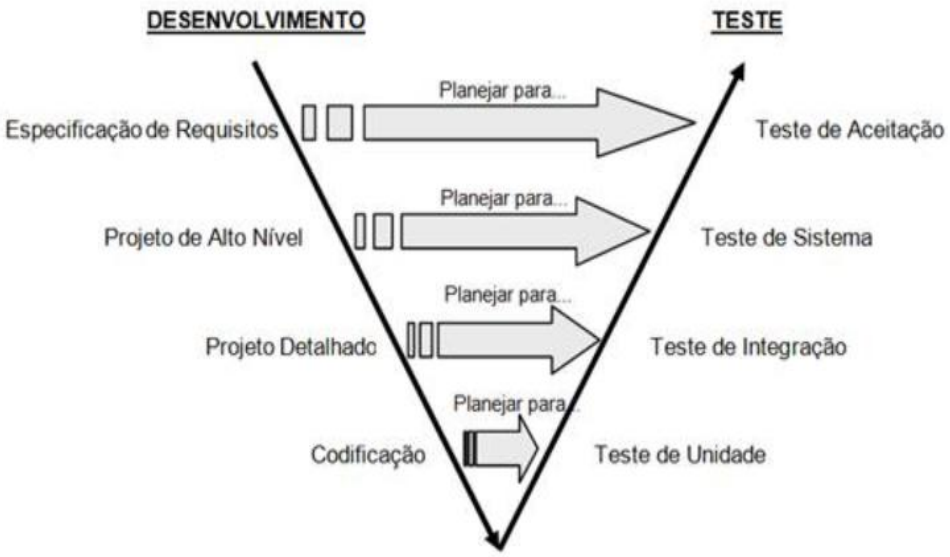

Figura 2 Modelo 'V' descrevendo o paralelismo entre as atividades de desenvolvimento e teste de *software*.

Fonte: (CRAIG, 2002).

2.2 Ferramentas de Teste de *Software*

As ferramentas de teste de *software* ajudam as equipes de desenvolvimento a investigarem erros no *software*, verificarem a funcionalidade e se certificarem a respeito da confiabilidade e da segurança do *software* desenvolvido. As ferramentas estão disponíveis para todos os estágios de um projeto de desenvolvimento de *software* (CENTERS, 2009).

Segundo Centers (2009), fatores determinantes e vantagens das Ferramentas de Teste de *Software* são:

- Aumentam a qualidade do *software* aplicativo;
- Avaliam o desempenho do *software*;
- Melhoram os processos de gerenciamento do ciclo de vida do produto;
- Conduzem análises comparativas e de risco;
- Ganham consistência nos procedimentos de teste com ferramentas automatizadas de teste (em contraste com os testes manuais que podem ser inconsistentes);
- Melhoram o tempo de comercialização como resultado de detecção eficaz de questões funcionais, de desempenho e de segurança;

- Economizam dinheiro com desenvolvimento de *software* e custos de manutenção;

Segundo o "*Guide to the CSTE Common Body of Knowledge*", apesar de não existir uma categorização amplamente difundida das ferramentas de teste, a experiência tem mostrado que elas são normalmente agrupadas em oito áreas distintas (CAETANO, 2007):

- Ferramentas de Automação de Testes de Regressão: essas ferramentas consistem na aplicação de teste à versão mais recente do *software*, para garantir que não surgiram novos defeitos em componentes já testados. Se, ao juntar um novo componente ou as suas alterações com os componentes restantes do sistema, surgirem novos defeitos em componentes inalterados, então se considera que o sistema regrediu. Estas ferramentas conseguem um resultado mais exato do teste executando exatamente os passos seguidos para o teste das primeiras versões já que elas permitem a gravação do teste. Exemplo: *Rational Functional Test* – IBM (proprietária) e *JUnit* – Java (*open source*) (WIKIPEDIA, 2011).
- Ferramentas para Gestão de Defeitos: através da gestão de defeitos podemos acompanhar a qualidade do *software* em teste com base nos defeitos cadastrados pelos testadores ao longo de um ciclo de testes. Essas ferramentas devem possuir um local aonde os testadores cadastram os *bugs* de forma organizada, devem possuir também um local para acompanhar o ciclo de vida dos defeitos e emitir relatórios de gestão. Exemplo: *Jira* (proprietária), *Bugzilla* e *Mantis* (*open source*) (DELFIM, 2008).
- Ferramentas para testes de Performance/*Stress*: o objetivo desse teste é caracterizar e avaliar o desempenho relacionado a característica do objetivo do teste, como perfis de andamento, fluxo de execução, tempos de resposta, confiabilidade e limites operacionais. Nas iterações de arquitetura, os testes de desempenho baseiam-se na identificação e na eliminação de gargalos de desempenho relacionados à arquitetura. Nas iterações de construção, outros tipos de testes de desempenho são implementados e executados para ajustar o *software* e o ambiente e para verificar se a aceitabilidade dos aplicativos e dos sistemas consegue lidar com condições de alta carga e *stress*, como um grande

número de transações, clientes e/ou volumes de dados. Exemplo: *WebLOAD* e *JMeter* (*open source*) (GUIDE, 2006).

- Ferramentas Manuais: uma das mais eficazes de todas as ferramentas de teste é uma simples *checklist* indicando quais itens que os testadores devem investigar e para permitir que os testadores assegurem que eles estão realizando as atividades de teste corretamente. Existem muitas ferramentas manuais, tais como tabelas de decisão, *scripts* de teste para ser utilizada para indicar operações de teste e *checklists* para testadores para usar ao executar tais técnicas de teste como revisões e inspeções (GUIDE, 2006).

- Ferramentas de Rastreabilidade: uma das ferramentas de rastreabilidade mais utilizadas é usada para traçar os requisitos desde o início do projeto através de operações (GUIDE, 2006).

- Ferramentas de Cobertura de Código: é uma ferramenta para calcular a porcentagem de código acessado pelos testes, além disso, ela calcula a porcentagem de caminhos possíveis abrangidos. Exemplo: *EMMA* (*open source*) (GUIDE, 2006).

- Ferramentas para Gestão de Testes*:* A gestão de testes é a parte principal de um processo de teste. A gestão é importante para o planejamento e controle das atividades de um projeto de teste. As ferramentas para gestão de testes devem oferecer um repositório central e padronizado aonde líderes de testes poderão criar *suítes* com os casos de teste, atribuir os casos de testes aos testadores, acompanhar o *status* da execução dos testes e emitir os relatórios com métricas e estatísticas. Exemplo: *Testlink*, *Testopia* (*open source*), *Rational Test Manager* (proprietária) (DELFIM, 2008).

- Ferramentas de Apoio à Execução dos Testes*:* Os testadores têm acesso a uma variedade de ferramentas de trabalho. Estas incluem o processamento de texto, planilhas, gráficos de computador utilizados para relatórios e verificação de *status*, e ferramentas que podem mensurar a dificuldade de leitura e documentação (GUIDE, 2006).

A maioria das organizações de teste concorda que, se as três seguintes diretrizes forem cumpridas, o uso das ferramentas serão mais eficazes e eficientes:

- Diretriz 1: Testadores não devem ser autorizados a utilizar ferramentas para as quais eles não receberam treinamento formal.
- Diretriz 2: O uso de ferramentas de teste deve ser incorporado aos processos de teste para que o uso das ferramentas seja obrigatório e não opcional.
- Diretriz 3: Testadores devem ter acesso a uma pessoa em sua organização, ou a organização que desenvolveu a ferramenta, para esclarecer dúvidas ou oferecer orientação sobre como usá-la (GUIDE, 2006).

2.3 Ferramentas Escolhidas

As ferramentas escolhidas para esse trabalho encontram-se na classificação de ferramentas para gestão de testes, sendo essas: *Testlink*, *Testopia* e *Rational TestManager*.

Primeiramente, seriam utilizadas apenas ferramentas *open source*. Entretanto, após a realização de uma pesquisa, verificou-se que várias ferramentas livres estavam defasadas, com documentação incompleta; sendo assim, optou-se pela escolha das duas ferramentas mais utilizadas para a gestão de testes: *Testlink* e *Testopia*. Além dessas, foi escolhida uma ferramenta proprietária: *Rational TestManager*, pois obteve-se mais facilmente o acesso a licença de utilização da mesma.

2.3.1 Testlink

O *Testlink* é uma ferramenta *open source* cujo principal objetivo é gerenciar a criação de Planos e Casos de Teste, possibilitando o controle dos testes planejados em relação aos testes executados. Permite associar um conjunto de Casos de Teste a um testador e acompanhar os resultados da execução dos testes. Oferece um recurso que possibilita registrar e organizar os requisitos do projeto, assim como associar os Casos de Teste aos requisitos (TEAMST, 2011). É desenvolvido usando plataformas como PHP e MYSQL. Por ser uma aplicação *WEB*, pode ser usado com qualquer servidor. Os conceitos e funcionalidades importantes presentes no *Testlink* incluem:

- Caso de Teste: descreve uma tarefa de testes via passos (ações, cenários) e resultados esperados. Os casos de testes são uma parte fundamental do *Testlink*. Os casos de teste constituem-se da seguinte maneira:

  - Título: pode incluir uma descrição breve ou abreviatura.

  - Sumário: deve ser realmente curto, apenas para uma visão geral.

- Etapas: descreve o cenário do teste (ações de entrada). Pode também incluir pré-condições e limpeza das informações.

- Resultados Esperados: descrevem a verificação e o comportamento esperados de um produto ou sistema testado.

- O número do ID de um caso de teste é atribuído, automaticamente pelo *Testlink* e não pode ser alterado pelos usuários. Este ID é um sistema amplo, o que significa que quando um caso de teste é criado, um contador global é utilizado, independente do projeto de teste no qual o caso de teste foi criado.

- Anexos: pode ser acrescentado se a configuração permitir.

- Palavras-chave: foram criadas para fornecer aos usuários outro nível de profundidade quando categorizados os casos de testes. Servem como um meio de agrupamento de casos de testes com alguns atributos dentro de uma especificação de teste.

- *Suíte* de Casos de Teste: organizam os casos de teste em unidades, o que estrutura a especificação dos testes em partes lógicas. Cada teste da *suíte* consiste de um título, descrição formatada dos casos de teste e, possivelmente, outros testes de *suítes*. *Testlink* utiliza a estrutura de árvore para teste de *suítes*. A prática comum é a de que a descrição detenha informação válida para a maioria dos dados incluídos. Por exemplo, o seguinte pode ser especificado: escopo, configuração, pré-condição, documentação relacionada, ferramentas, infraestrutura, etc. Os usuários (com direitos de edição) podem criar, apagar, copiar, mover, exportar e importar ambos os testes de *suítes* e casos de testes aninhados. Títulos e descrição podem ser modificados também. Anexos com documentos ou imagens poderão ser adicionados em teste de *suítes* particulares.

- Planos de Testes: podem ser inventados a partir dos casos de testes de um ou muitos projetos de testes. Os planos de testes contêm nome, descrição, coleção de casos de testes escolhidos, *builds*, resultados dos testes, marcos, testador e sessão de definição de prioridade.

    - Criar e apagar Planos de Testes: Planos de teste podem ser criados por usuários com privilégios de líder para os atuais projetos de testes. São compostos de casos de testes importados de uma especificação de testes em um ponto específico de tempo. Podem ser excluídos pelos usuários com privilégios de líderes. Excluindo planos de testes permanentemente, apagará tanto o plano de teste e todos os seus dados correspondentes, incluindo os casos de testes, resultados, etc.

- *Builds*: é um *release* específico do *software*. Cada projeto em uma companhia é provavelmente feito de muitos diferentes *builds*. A execução do *Testlink* é feita de *builds* e casos de testes. Se não existirem *builds* criados para um projeto, a tela não permitirá executá-lo. A tela de métrica também ficará completamente branca.
- Povoando planos de teste – Adicionando casos de Testes: os dados de múltiplos projetos de testes podem ser adicionados em um plano de teste. Os dados da especificação de teste podem ser filtrados por palavras-chave. Uma vez que os dados tiverem sido ligados a um plano de teste, ele será marcado com uma marca de checagem. Se um caso de teste já tiver sido importado ele será ignorado se for importado de novo.

- Projeto de Teste: é a base organizacional da unidade de *Testlink*. Os projetos de teste podem alterar as suas características e funcionalidades ao longo do tempo, mas, na maior parte dos casos, continuam sendo os mesmos. O projeto de teste inclui requisitos de documentação, especificação de testes, planos de testes e direitos específicos dos usuários.
- Requisitos: os requisitos estão agrupados por documentos de especificação de requisitos, os quais estão relacionados ao projeto de testes. O *Testlink* não suporta (ainda) versões para especificação de requisitos e requisitos. Logo, a versão do documento deve ser inserida após o título da especificação. Um usuário pode inserir uma descrição simples ou notas no campo escopo.
- Execução de Teste: a execução do teste está disponível quando:

  a) A especificação do teste está escrita.

  b) O plano de teste é criado.

  c) Os casos de testes são adicionados ao plano de testes.

  d) Um *build* é criado.

  e) O plano de teste é nomeado aos testadores.

- Relatórios de Testes e Métricas: os relatórios de testes e métricas são acessados clicando em “Resultados” ou “Relatórios de testes e métricas” nos links na página principal. Relatórios e métricas baseiam-se no plano de teste selecionado. A página que é mostrada ao usuário inclui: o painel direito será preenchido com instruções sobre como usar os controles e como cada relatório é produzido. O painel esquerdo é usado para navegar por

cada relatório e por controles operacionais que controlam o efeito e o comportamento dos relatórios que são mostrados.

- Administração do Usuário: cada usuário do sistema será capaz de editar suas próprias informações através do *link* "Pessoal" no menu superior. O *Testlink* permite usuários, com direitos de administrador, de criar, editar e excluir usuários dentro do sistema. No entanto, não permite que os administradores visualizem ou editem senhas do usuário. Se os usuários esquecem suas senhas, há um *link,* na tela de *login*, que irá enviar suas senhas utilizadas em seu nome de usuário e endereço de *e-mail* que entrou.

O *Testlink* é construído com seis diferentes níveis de permissão padrões. Estes níveis de permissão são os seguintes:

- ***Guest*:** só tem permissão para visualizar casos de teste e métricas do projeto.
- ***Tester*:** só pode documentar a execução dos casos de teste.
- ***Test Senior*:** pode documentar a execução de casos de teste, manipular a área de *test specification* criando *suítes* de teste ou casos de teste.
- ***Leader*:** possui permissões para gerenciar planos de teste, criar *builds*, definir prioridades e todas as permissões dos usuários *Guest*, *Tester* e *Test Senior*.
- ***Administrator*:** possui todas as permissões dentro do *Testlink*.
- ***Test Designer*:** possui permissões para gerenciar a área de *test specification*, criando *suítes* de teste e casos de teste, porém não pode executar casos de teste (FREIRE, 2008).

Na Figura 3 está ilustrada a tela principal do *Testlink*.

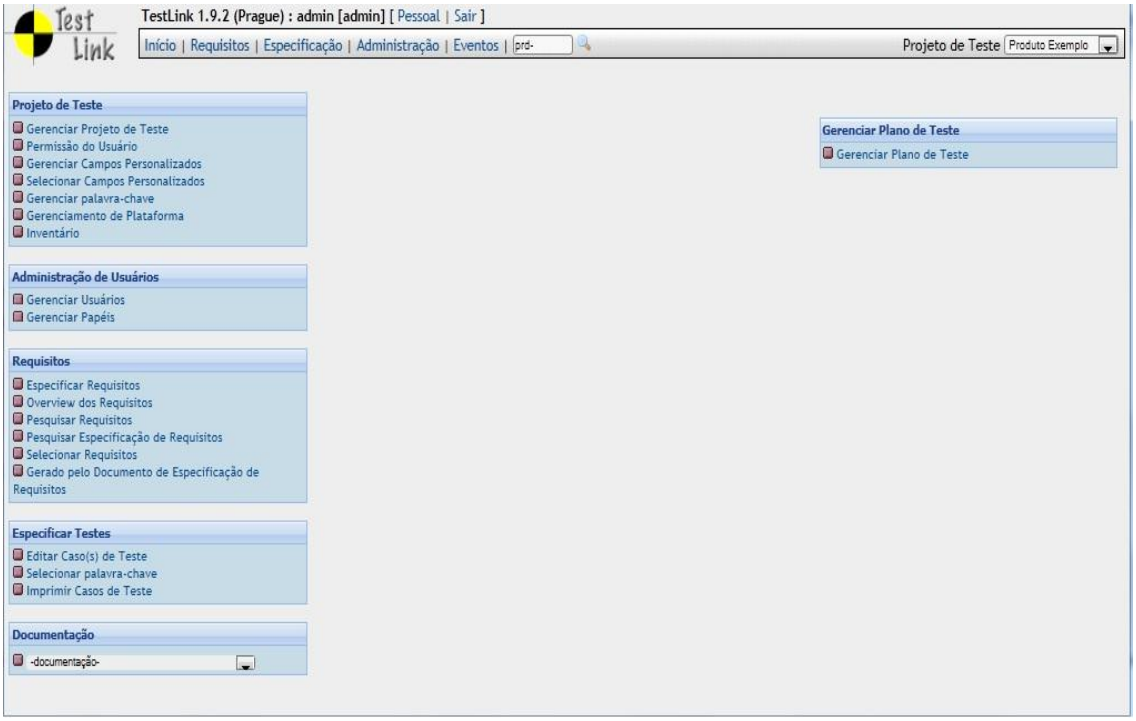

Figura 3 Tela Principal do *Testlink*.

Fonte: (TEAMST, 2011).

2.3.2 Testopia

Testopia é uma extensão de gestão de casos de teste para o *Bugzilla*[1]. Ele foi projetado para ser uma ferramenta genérica para monitoramento de casos de teste, permitindo uma organização dos testes para integração de relatórios de *bugs* com os resultados da execução dos casos de teste (TESTOPIA, 2010). Embora o *Testopia* tenha sido projetado principalmente para teste de *software*, ele pode ser usado para controlar qualquer tipo de casos de teste (HENDRICKS, 2010).

O *Testopia* foi desenvolvido para fornecer um repositório central para os esforços colaborativos de distribuição de testadores. Ele serve como um repositório de casos de teste e gerenciamento do sistema; é projetado para atender às necessidades dos testadores de *software* de todas as dimensões de grupos e organizações.

O *Testopia* é executado em *Linux* e oficialmente não suporta a instalação no *Windows* (MOZZILA, 2010).

A integração entre o *Bugzilla* e o *Testopia* é representada pela Figura 4, onde são mostradas as ligações e os componentes das mesmas.

[1] *Bugzilla* é uma ferramenta baseada em *web* e *e-mail* que fornece suporte ao desenvolvimento do projeto *Mozilla*, rastreando defeitos e servindo também como plataforma para pedidos de recursos.

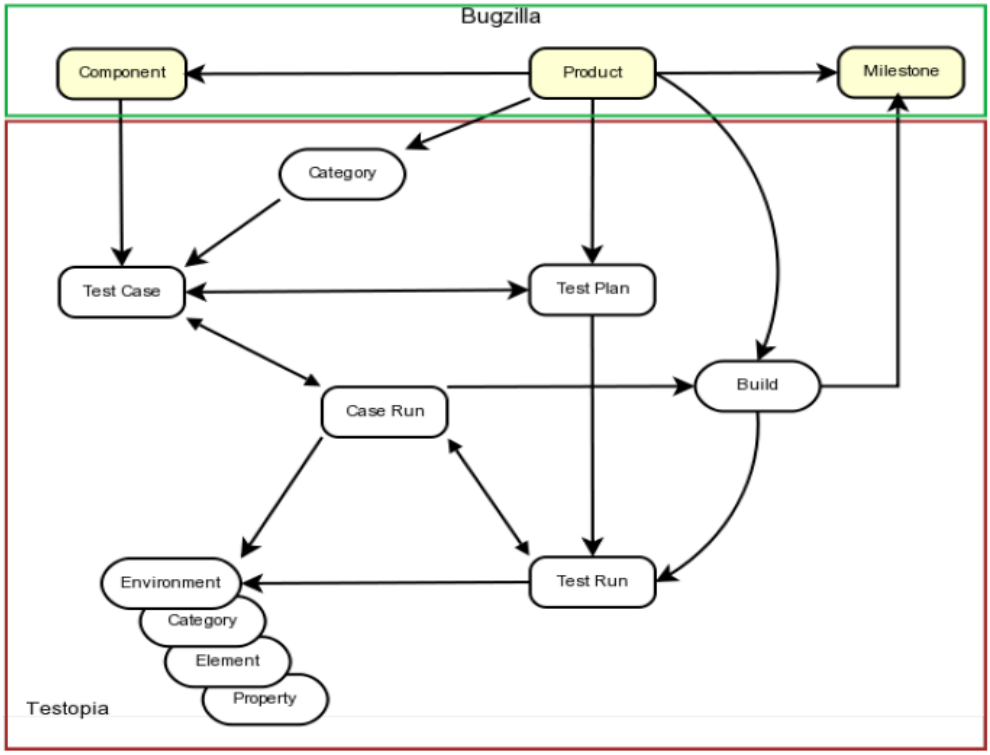

Figura 4 Integração *Bugzilla* e *Testopia*.
Fonte: (HENDRICKS, 2010).

O *Testopia* é uma extensão para o *Bugzilla*, baseia-se em muitos dos mesmos objetos que o *Bugzilla* usa para rastrear *bugs*. Pata utilizar o *Testopia*, primeiro configuram-se os produtos e as versões usando o editor de produto do *Bugzilla*.

O *Testopia* é composto de vários objetos que são interdependentes. Juntos, eles fazem a gestão do processo de teste possível. Abaixo estão os elementos que fazem parte:

- Plano de Teste: no topo da hierarquia do *Testopia* estão os planos de teste. Antes de fazer qualquer outra coisa, precisa-se criar um plano de teste. Eles estão associados a um produto único no *Bugzilla*, mas podem-se ter vários planos de teste para cada produto. O plano servirá como ponto de armazenamento para todos os casos de teste e execuções de teste relacionado e atuará como o painel de controle para os testes. Também servirá para determinar quem terá acesso à atualização de casos de teste.
- Casos de Teste: os casos de teste são a base de todos os testes; mostram quais as medidas tomar durante a execução de um teste e que resultados esperar. Se uma execução particular de teste através das etapas não produzir o resultado esperado, o teste falha. Os casos de teste são semi-independentes no *Testopia*. Cada caso de teste pode ser associado com múltiplos planos de teste, mas alguns cuidados devem ser tomados para garantir a

atualização de um caso de teste para não interferir com os testes em outro plano de teste. Uma lista de planos de teste associados é exibida com cada caso de teste.

- Categorias: os casos de teste podem ser divididos em categorias. Podem-se definir muitas categorias para o produto, a que o testador desejar da página de plano de testes. Estas categorias, no entanto, não podem ser confundidas com componentes. Cada produto no *Bugzilla* é dividido em componentes e podem-se aplicar vários componentes para cada caso de teste, porém cada caso de teste só pode pertencer a uma categoria de cada vez.
- *Test Run*: *test runs* são os focos principais do esforço de teste. Depois de ter definido um conjunto de casos de teste, deve-se executar os testes em um *test run*. Cada execução é associada a um único plano de testes e pode consistir em qualquer número de casos de teste a partir desse plano.
- *Test Run Environments*: se os casos de teste são o "o quê" dos testes, então ambientes são "onde". Nenhum teste é executado no vácuo. Onde o teste é executado é tão importante quanto executar. O *software* geralmente é projetado para ser executado em um hardware específico e sob condições específicas. Estas condições são capturadas no ambiente de teste. Ambientes são aplicados para executar testes diretamente, mas podem ser aplicados a casos de teste indiretamente.
- *Builds*: no *Testopia*, cada iteração é chamada de *build*. *Builds* estão frequentemente associados a marcos do projeto. Independentemente de se usar marcos no *Bugzilla*, precisa-se definir pelo menos uma *build* para o produto antes de começar a executar o teste.
- *Test Case-Runs*: uma execução de um caso de teste é o registro de como um caso de teste particular sairá em uma execução particular para um determinado *build* em um determinado ambiente. Quando se cria um *test run*, os registros de cada caso de teste que executam são criados. Por padrão esses assumem a *build* e ambiente de execução do teste, no entanto, é possível alterar esses atributos em um determinado processo de execução, essencialmente criando um novo processo de execução para cada combinação. Isso é desejável em situações em que a maioria dos casos de teste são bastante genéricos no escopo do ambiente, mas os casos de teste simples podem exigir condições específicas (MOZZILA, 2010).

Na Figura 5, tem-se a tela principal do *Testopia*, na qual é possível verificar que esta ferramenta é um complemento da ferramenta de *bugs*, *Bugzilla*.

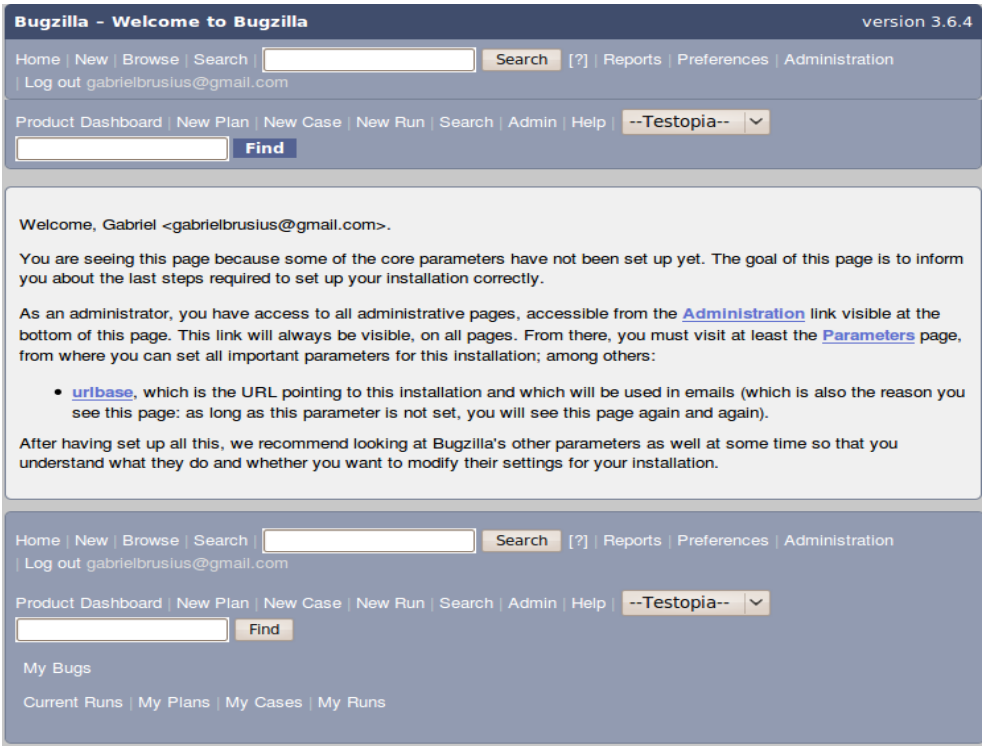

Figura 5 Tela Principal do *Testopia*.

Fonte: (TESTOPIA, 2011).

2.3.3 Rational TestManager

Rational TestManager é um *framework* que reúne ferramentas, propriedades e dados envolvidos na atividade de teste. Ele suporta as cinco principais etapas de teste definidas no Processo Unificado da *Rational* (RUP) (SILVESTRI, 2003):

1) Planejamento
2) Projeto
3) Implementação
4) Execução
5) Avaliação

1. Planejamento: a atividade de planejamento de testes consiste, basicamente, em responder à pergunta "O que eu preciso testar?". Ao terminar o planejamento do teste, tem-se um plano de teste que define o que será testado.

No *Rational TestManager*, um plano de teste pode ter muitas propriedades. Elas podem ser adicionadas na criação do plano de teste ou posteriormente e podem ser alteradas. Algumas propriedades são:

- Uma descrição do plano de teste;
- O proprietário do plano de teste;
- As iterações e configurações associadas ao plano de teste;
- Os documentos externos associados ao plano de teste.

No *Rational TestManager*, um plano de teste pode conter uma lista de casos de teste. Os casos de teste podem ser organizados com base nas pastas de casos de teste. Após planejar os testes, pode-se projetá-los.

2. Projeto: a atividade de projeto de teste consiste, basicamente, em responder à pergunta, "Como eu vou fazer um teste?". Ao terminar o projeto de teste, tem-se um *design* de teste que ajudará a compreender como será realizado o caso de teste e também a começar o planejamento da implementação.

No *Rational TestManager*, pode-se projetar os casos de teste indicando os passos reais que precisam ocorrer no teste. Também pode especificar pré-condições, pós-condições e critérios de aceitação.

Após projetar os testes, pode-se implementá-los.

3. Implementação: a atividade de implementação de teste consiste, basicamente, em criar *scripts* de teste reutilizáveis.

No *Rational TestManager*, pode-se implementar os testes criando *scripts* manuais. Um *script* manual é um conjunto de instruções de teste que serão executadas por um testador humano. Pode-se implementar testes automatizados usando o *Rational Robot*.

Após implementar os testes, pode-se executá-los no *Rational TestManager*.

4. Execução: a atividade de execução de teste consiste, basicamente, em executar os *scripts* de teste para certificar-se de que o sistema funciona corretamente.

No *Rational TestManager*, pode-se executar os testes de diversas maneiras:

- Executar um *script* de teste individual, que executa uma única implementação.
- Executar um ou mais casos de teste, que executa as implementações dos casos de teste.
- Executar uma *suíte*, que executa os casos de teste e suas implementações através de vários computadores e usuários.

Após executar os testes, podem-se avaliar os resultados.

5. Avaliação: a atividade de avaliação de teste consiste em determinar a qualidade do sistema em teste.

No *Rational TestManager*, pode-se avaliar os testes examinando os resultados da execução no *log* e gerando diversos relatórios.

O *log* do teste indica se houve falha na execução do *script* e permite a obtenção de informações detalhadas necessárias para a avaliação dos resultados. A partir dele, pode-se identificar e registrar as solicitações de mudança.

Há três tipos básicos de relatórios no *Rational TestManager:*

- Relatórios de tendência e de distribuição do Caso de Teste: ajudam a acompanhar o planejamento, a implementação e os resultados da execução do caso de teste.
- Relatórios de Teste de Desempenho: ajudam a avaliar a eficiência relativa com a qual um aplicativo realiza tarefas-chave sob determinadas condições.
- Relatórios de listagem: exibem listas dos diferentes componentes do teste armazenados em um projeto da *Rational* (CORPORATION, 2001).

Na Figura 6, tem-se a tela principal do *Rational TestManager*.

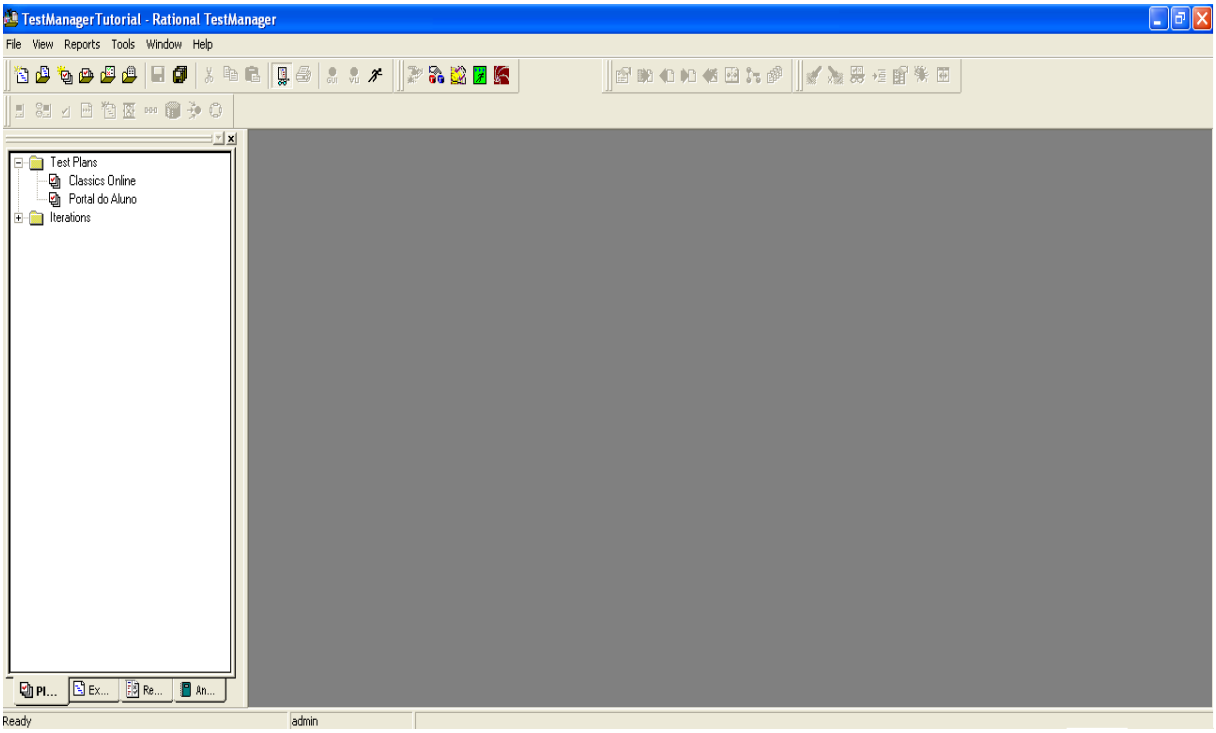

Figura 6 Tela Principal do *Rational TestManager*

Fonte: (TESTMANAGER, 2011).

2.4 Critérios de Avaliação de Ferramentas de Teste

Critérios de avaliação de ferramentas de testes podem ser utilizados para definir a qualidade e/ou as funcionalidades fornecidas pela ferramenta analisada. Baseando-se no artigo '*Criteria for Software Testing Tool Evaluation – A Task Oriented View*' (ILLES, 2005), identificam-se atividades que potencialmente podem ser automatizadas; através dessas atividades derivam-se critérios de avaliação para ferramentas de teste.

Ao consultar a internet, pode-se encontrar uma imensa variedade de ferramentas de apoio às diversas atividades realizadas durante o processo de teste (*JUnit*, *Bugzilla*, *JMeter*, *EMMA*, *Testlink*, etc). Embora tenha uma inúmera variedade dessas ferramentas, não é fácil identificar a ferramenta mais adequada para uso em um determinado projeto.

3 MATERIAIS E MÉTODOS

Neste capítulo é apresentado o método científico utilizado na elaboração deste trabalho. Primeiramente, foi realizada uma pesquisa bibliográfica envolvendo os seguintes assuntos: testes de *software*, ferramentas de testes de *software*, aspectos importantes sobre as ferramentas de gestão de teste selecionadas, e critérios para avaliação de ferramentas de gestão de testes.

Após isso, foi realizado um experimento no qual as ferramentas foram instaladas e os critérios de avaliação foram aplicados nas três ferramentas escolhidas, por meio da realização de um estudo de caso visando analisar cada atividade envolvida no processo de teste. As ferramentas de gerência de teste utilizadas foram: *Testlink*, *Testopia* e *Rational TestManager*, descritas no capítulo 2.

Anteriormente à aplicação dos critérios de avaliação, foram realizados casos de teste, para possível aplicação desses critérios, e análise do suporte das ferramentas. Foi utilizado o recurso 'Portal do Aluno' no site da Universidade Federal de Santa Maria, para serem desenvolvidos os casos de teste (APÊNDICE A). As funcionalidades utilizadas nesse recurso foram: fazer *login*, matricular disciplina, consultar horário, imprimir horário, consultar histórico, imprimir histórico e mudar senha; as quais foram aplicadas nos critérios de avaliação, testando cada funcionalidade para cada critério.

Os critérios de avaliação das ferramentas foram baseados nas atividades do processo tradicional de teste, conforme é projetado na Tabela 1.

Tabela 1: Tarefas do processo de teste

ID	TAREFAS
A	Projetando Casos de Teste
B	Construindo Casos de Teste
C	Executando casos de teste
D	Acompanhamento dos Resultados da Execução dos Testes
E	Relatórios de resultados de teste

Fonte: *Criteria for Software Testing Tool Evaluation – A Task Oriented View*, 2005

A seguir é dada uma definição dos critérios de teste definidos para serem aplicados durante o experimento nas três ferramentas de teste selecionadas. Os critérios de teste foram definidos com base em (ILLES, 2005 e GUIDE, 2006):

A) Projetando Casos de Teste - as principais atividades para projetar casos de teste são: selecionar as técnicas de teste, criar dependência entre os casos de teste, importar requisitos, testar a qualidade da aplicação, organizar os casos de teste em *suítes* e classificá-los de acordo com a prioridade.

   As ferramentas fornecem suporte para:

   1. Selecionar técnicas de teste (caixa-preta e/ou caixa-branca).
   2. Criar dependência entre os casos de teste.
   3. Importar requisitos para os casos de teste em diferentes extensões.
   4. Utilizar casos de teste para testar os critérios de qualidade da aplicação.
   5. Organizar os casos de teste em *suítes* para um melhor gerenciamento dos testes.
   6. Classificar os casos de teste de acordo com a prioridade dos mesmos.

B) Construindo Casos de Teste: a principal atividade de construção de casos de teste inclui: a implementação de casos de teste. No caso de automação de testes, *scripts* de teste são desenvolvidos.

   As ferramentas fornecem suporte para:

      1. Editar *scripts* de teste.
      2. Desenvolver código de teste em conformidade com as práticas de engenharia de *software*.
      3. Capturar casos de teste executáveis.
      4. Gerar dados de teste (in) válidos.

C) Executando Casos de Teste: os casos de teste projetados e construídos nas fases anteriores devem ser executados. Essas atividades podem ser feitas manualmente (executando os passos definidos do caso de teste) ou em modo automatizado executando o *script* de teste.

   As ferramentas fornecem suporte para:

   1. Possuir um filtro de informações para otimizar a procura e execução de testes.

2. Suportar relatórios de *bugs.*
3. Continuar a execução de um teste suspendido.
4. Escolher entre executar a *suíte* de testes ou somente casos de teste separados.

D) Acompanhamento dos Resultados da Execução dos Testes: a principal atividade de acompanhamento dos resultados da execução dos testes inclui: acompanhamento em tempo real da execução dos testes, para uma análise mais completa da execução.

1. Acompanhar a execução dos testes em tempo real.

E) Relatórios de Resultados de Teste: relatórios de atividade visam agregar informações de teste em diferentes níveis de detalhe em uma forma compreensível e reprodutível. Portanto, os relatórios devem ter um grau de detalhe personalizável com relação ao público alvo (gestores, clientes e desenvolvedores).

As ferramentas fornecem suporte para:

1. Emitir relatórios.
2. Configurar o ambiente para gerar relatórios por papéis.
3. Gerar relatórios em diferentes extensões (Pdf, Word, Excel, HTML).

Por fim, foi realizada uma análise dos resultados obtidos, através dos critérios de avaliação das ferramentas e uma comparação entre as ferramentas utilizadas.

4 RESULTADOS

Esse capítulo descreve a análise das ferramentas de gestão de testes segundo os critérios definidos e descritos no capítulo 3. Foram identificadas as atividades realizadas e os critérios resultantes foram projetados de forma que pudessem ser avaliados pela presença (sim) ou ausência (não) de determinada funcionalidade/atividade, para permitir uma avaliação eficiente de uma variedade de ferramentas de teste.

4.1 Projetando Casos de Teste

A Tabela 2 mostra o resultado da análise desse critério nas ferramentas selecionadas. Tabela 2: Projetando Casos de Teste

			FERRAMENTAS		
ATIVIDADES	**ID**	**CRITÉRIOS**	***TESTLINK***	***TESTOPIA***	***TESTMANAGER***
Projetando Casos de Teste	1	Selecionar as técnicas de teste (caixa-branca, caixa-preta)	Não	Não	Não
	2	Criar dependências entre os casos de teste	Não	Sim	Não
	3	Importar requisitos de diferentes extensões para os casos de teste	Sim	Não	Sim
	4	Utilizar casos de teste para testar os critérios de qualidade da aplicação	Não	Não	Não
	5	Organizar os casos de teste em *suítes* para um melhor gerenciamento dos testes	Sim	Sim	Sim
	6	Classificar os casos de teste de acordo com a prioridade dos mesmos	Sim	Sim	Não

A seguir é dada uma definição dos critérios utilizados para analisar a atividade Projetando Casos de Teste. Para os critérios serem válidos foram considerados os seguintes requisitos:

- **Critério 1**: A ferramenta deve ter suporte para a ação de selecionar as técnicas de teste, ou seja, escolher entre teste caixa-preta e/ou caixa-branca.
- **Critério 2**: A ferramenta deve ter suporte a uma funcionalidade para a criação de dependências entre os casos de teste. Exemplo: se o caso de teste T2 for dependente do caso de teste T1, ele poderá ser executado somente após a execução de T1.
- **Critério 3**: A ferramenta deve ter suporte à importação de requisitos para os casos de teste em diferentes extensões. Com isso, a mesma apresenta uma maior diversidade, gerando assim, mais alternativas para projetar os casos de teste.
- **Critério 4**: A ferramenta deve ter suporte para realizar testes e analisar os critérios da qualidade da aplicação.
- **Critério 5**: A ferramenta deve ter suporte à criação de *suítes* de teste. Uma *suíte* testes organiza os casos de testes em unidades, o que estrutura a especificação dos testes em partes lógicas.
- **Critério 6**: Na criação dos casos de teste, a ferramenta deve ter suporte a uma funcionalidade onde pode-se destacar a prioridade dos casos de teste. Onde o caso de teste com a prioridade mais alta deve ser executado antes que o de prioridade mais baixa.

4.1.1 FERRAMENTA *TESTLINK*:

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testlink* não possui suporte para este critério. É possível projetar os casos de teste, entretanto, essa ferramenta não se preocupa com a técnica de teste a ser utilizada (caixa-preta ou caixa-branca).
- **Critério 2**: A ferramenta *Testlink* não possui suporte para este critério, ou seja, se um caso de teste for dependente de outro, deve-se especificar no mesmo a sua dependência, para o testador executar um caso de teste primeiramente e após, o outro.
- **Critério 3**: O *Testlink* suporta a importação de requisitos em XML e CSV. Pode-se criar uma especificação de requisitos na própria ferramenta. Sendo assim, têm-se duas formas de especificar requisitos.

- **Critério 4**: O *Testlink* não suporta a análise da qualidade da aplicação realizada pelos testes. Para isso, existem ferramentas específicas para a execução e análise de desempenho do objetivo do teste.
- **Critério 5**: O *Testlink* possui a opção de criação de *suítes* de teste. Cada *suíte* possui um nome, palavras-chave (opcional) e detalhes (o que essa *suíte* contém). As *suítes* também podem ser importadas em formato XML; também funcionam como um diretório, assim os usuários podem mover e copiar *suítes* de testes dentro do projeto de testes.
- **Critério 6**: No *Testlink* pode-se classificar os casos de teste de acordo com a prioridade. Ele dispõe de três tipos de prioridade: baixa, média e alta.

4.1.2 FERRAMENTA *TESTOPIA*:

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testopia* não possui suporte para este critério. É possível projetar os casos de teste, entretanto, essa ferramenta não se preocupa com a técnica de teste a ser utilizada (caixa-preta ou caixa-branca).
- **Critério 2**: A ferramenta *Testopia* possui suporte a esse critério. Ela apresenta uma funcionalidade nos casos de teste chamada "Dependências", contendo duas caixas de texto, uma chamada "*Blocks*" e outra "*Depends On*", onde inserem-se os casos de teste que dependem um do outro.
- **Critério 3**: A ferramenta *Testopia* não oferece suporte a importação de requisitos. Somente pode-se criar os requisitos na própria ferramenta.
- **Critério 4**: O *Testopia* não suporta a análise da qualidade da aplicação realizada pelos testes. Para isso, existem ferramentas específicas para a execução e análise de desempenho do objetivo do teste.
- **Critério 5**: O *Testopia* também tem a opção de criação de *suítes* de teste. Nesta ferramenta apenas o nome *suíte* é trocado por categorias, mas segue a mesma ideia das *suítes* de teste.
- **Critério 6**: No *Testopia* também pode-se classificar os casos de teste de acordo com a prioridade. Ele possui cinco tipos de prioridade: P1, P2, P3, P4, P5. Onde P1 é a prioridade mais baixa e P5 a prioridade mais alta.

4.1.3 FERRAMENTA *TESTMANAGER*:

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *TestManager* não possui suporte para este critério. É possível projetar os casos de teste, entretanto, essa ferramenta não se preocupa com a técnica de teste a ser utilizada (caixa-preta ou caixa-branca).
- **Critério 2**: A ferramenta *TestManager* não possui suporte a esse critério. Sendo assim, deve-se especificar no caso de teste se o mesmo apresenta dependência ou não.
- **Critério 3**: No *TestManager* pode-se importar os requisitos de três modos diferentes: pelo Excel, e por mais duas ferramentas específicas de requisitos da *Rational*: o *Rational RequisitePro* e o *Rational Rose*. Os requisitos só podem ser importados, ou seja, não é possível especificar os requisitos se não for nesses três métodos citados.
- **Critério 4**: O *TestManager* não suporta a análise da qualidade da aplicação realizada pelos testes. Para isso, existem ferramentas específicas para a execução e análise de desempenho do objetivo do teste.
- **Critério 5**: O *TestManager* possui suporte também a criação de *suítes* de teste.
- **Critério 6**: O *TestManager* não possui a funcionalidade de prioridade dos casos de teste. Assim, deve-se apontar no próprio caso de teste, a prioridade do mesmo.

4.2 Construindo Casos de Teste

A Tabela 3 mostra o resultado da análise desse critério nas ferramentas selecionadas.

Tabela 3: Construindo Casos de Teste

			FERRAMENTAS		
ATIVIDADES	**ID**	**CRITÉRIOS**	***TESTLINK***	***TESTOPIA***	***TESTMANAGER***
Construindo Casos de Teste	1	Editar *scripts* de testes	Não	Não	Sim
	2	Desenvolver código de teste em conformidade com as práticas de engenharia de *software*	Não	Não	Não
	3	Capturar casos de teste executáveis	Não	Não	Não
	4	Gerar dados de teste (in) válidos	Não	Não	Não

A seguir é dada uma definição dos critérios utilizados para analisar a atividade Construindo Casos de Teste. Para os critérios serem válidos foram considerados os seguintes requisitos:

- **Critério 1**: A ferramenta deve ter suporte a edição de *scripts* de teste manual ou automatizado. Edita-se um *script* de teste geralmente para adicionar um novo recurso a um *script* existente, inserir uma nova sequência de ação, entre outras funções.
- **Critério 2**: A ferramenta de teste deve possuir suporte ao desenvolvimento de código de teste. A ferramenta também deve respeitar o desenvolvimento de acordo com as práticas de Engenharia de *Software*, como por exemplo, modularização do código, comentários, declaração de variáveis e tipos de dados; passagem de parâmetro, separação de dados de teste e a definição do rumo dos acontecimentos.
- **Critério 3**: A ferramenta deve ter suporte a um mecanismo de captura de casos de teste executáveis, para a edição e execução dos mesmos.
- **Critério 4**: A ferramenta deve ter suporte para construção de casos de teste e geração dos dados de testes, especificando os mesmos em válidos ou inválidos, através da análise dos dados gerados.

4.2.1 FERRAMENTA *TESTLINK*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testlink* não possui suporte para este critério, ou seja, não edita *scripts* de teste, pois não automatiza casos de teste, somente executa testes manuais.
- **Critério 2**: O *Testlink* não possui suporte a esse critério, pois não realiza teste automatizado, somente armazena o resultado da execução manual.
- **Critério 3**: O *Testlink* não possui suporte a esse critério. Atualmente, existem ferramentas específicas de captura e reprodução que suportam esse mecanismo.
- **Critério 4**: O *Testlink* não possui suporte a esse critério, pois não realiza a geração dos dados de teste, apenas classifica-os como válidos ou inválidos.

4.2.2 FERRAMENTA *TESTOPIA*:

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testopia* não possui suporte a esse critério, ou seja, não edita *scripts* de teste, pois não automatiza casos de teste.
- **Critério 2**: O *Testopia* não possui suporte a esse critério, pois não realiza teste automatizado, somente armazena o resultado da execução manual.
- **Critério 3**: O *Testopia* não possui suporte a esse critério. Atualmente, existem ferramentas específicas de captura e reprodução que suportam esse mecanismo.
- **Critério 4**: O *Testopia* não possui suporte a esse critério, pois não realiza a geração dos dados de teste, apenas classifica-os como válidos ou inválidos.

4.2.3 FERRAMENTA *TESTMANAGER*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *TestManager* possui suporte a esse critério. Ela consegue editar *scripts* de teste manuais no *Rational ManualTest*.
- **Critério 2**: O *TestManager* não possui suporte a esse critério, pois na *suíte* de ferramentas utilizada para este estudo, não são realizados testes automatizados.
- **Critério 3**: O *TestManager* não possui suporte a esse critério. Atualmente, existem ferramentas específicas de captura e reprodução que suportam esse mecanismo.

- **Critério 4**: O *TestManager* não possui suporte a esse critério, pois não realiza a geração dos dados de teste, apenas classifica-os como válidos ou inválidos.

4.3 Executando Casos de Teste

A Tabela 4 mostra o resultado da análise desse critério nas ferramentas selecionadas.

Tabela 4: Executando Casos de Teste

			FERRAMENTAS		
ATIVIDADES	**ID**	**CRITÉRIOS**	***TESTLINK***	***TESTOPIA***	***TESTMANAGER***
Executando Casos de Teste	1	Possuir um filtro de informações para otimizar a procura e execução de testes	Sim	Sim	Não
	2	Suportar relatório de *bugs*	Não	Sim	Não
	3	Continuar a execução de um caso de teste suspendido	Sim	Sim	Sim
	4	Escolher entre executar a suíte de testes ou somente casos de teste separados	Não	Não	Sim

A seguir é dada uma definição dos critérios utilizados para analisar a atividade Executando Casos de Teste. Para os critérios serem válidos foram considerados os seguintes requisitos:

- **Critério 1**: A ferramenta deve ter suporte a um filtro de buscas de casos de teste. De acordo com essa busca, a ferramenta irá mostrar apenas os testes que realmente o usuário quer testar.
- **Critério 2**: A ferramenta deve ter suporte a uma funcionalidade de cadastro de *bugs*.

- **Critério 3**: A ferramenta deve ter suporte ao suspender um caso de teste, continuá-lo após a interrupção do mesmo. Exemplo: um caso de teste é suspenso, entretanto, é possível retornar ao estado no qual ele se encontrava, antes da interrupção.
- **Critério 4**: A ferramenta deve ter suporte a execução de somente um caso de teste por vez, ou testar uma *suíte* de teste, a qual testa vários casos de teste em sequência.

4.3.1 FERRAMENTA *TESTLINK*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testlink* possui suporte para este critério. Ela possui um filtro de busca, o qual contém inúmeros campos (ID do Caso de Teste, Título do Caso de Teste, *Suíte* de Teste, Palavra-chave, Prioridade, Tipo de Execução, Atribuído para, Resultado, *Build*) para se realizar uma execução dos casos de teste de acordo com a busca realizada. Otimizando assim, a execução dos casos de teste.
- **Critério 2**: A ferramenta *Testlink* não possui suporte para este critério. Entretanto, integrada a outras ferramentas de Gestão de Defeitos (*Bugzilla*, *Mantis*, *Jira*) é possível ter suporte para esse critério.
- **Critério 3**: A ferramenta *Testlink* possui suporte para este critério. Pode-se continuar a execução de um caso de teste suspenso. Por padrão o *Testlink* coloca o caso de teste como "Não Executado" antes da execução. Após a mudança desse estado para qualquer outro (Passou, Com Falha, Bloqueado) não podemos mudar o estado para "Não Executado" novamente e nem podemos editar os casos de teste que já foram executados.
- **Critério 4**: A ferramenta *Testlink* não possui suporte a esse critério, ou seja, por padrão quando cria-se um caso de teste, primeiro tem que criar uma *suíte* de teste, a qual pode conter apenas um ou vários casos de teste. Entretanto, a execução dos casos de teste será feito "um por um". Não consegue-se selecionar uma *suíte* de teste e executar os casos de teste sequencialmente.

4.3.2 FERRAMENTA *TESTOPIA*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testopia* possui suporte a esse critério. Ela possui um filtro de busca o qual contém inúmeros campos (*Status*, Categoria, *Build*, Ambiente, Prioridade, Tipo de Teste, entre outros) para se realizar uma execução dos casos de teste de acordo com a busca realizada.
- **Critério 2**: A ferramenta *Testopia* possui suporte a esse critério, pois é classificada como uma extensão da ferramenta de gestão de defeitos *Bugzilla*.
- **Critério 3**: A ferramenta *Testopia* possui suporte a esse critério. Pode-se continuar a execução de um caso de teste suspenso. No *Testopia*, por padrão, o teste é marcado como "*Idle*", ou seja, o caso de teste ainda não foi examinado.
- **Critério 4**: A ferramenta *Testopia* não possui suporte a esse critério, ou seja, quando cria-se os casos de teste, obrigatoriamente adiciona-se a uma categoria (*suíte*). A execução dos casos de teste será feito "um por um". Não consegue-se selecionar uma categoria de teste e executar os casos de teste sequencialmente. As categorias servem basicamente para organizar os casos de teste.

4.3.3 FERRAMENTA *TESTMANAGER*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *TestManager* não possui suporte a esse critério. Para executar um caso de teste específico, deve-se procurar manualmente por ele, sem ajuda de filtros.
- **Critério 2**: A ferramenta *TestManager* não possui suporte para este critério. Entretanto, ela pode ser integrada a uma ferramenta de Gestão de Defeitos da *Rational*, chamada *Rational ClearQuest*, sendo assim possível reportar *bugs*.
- **Critério 3**: A ferramenta *TestManager* possui suporte a esse critério. Pode-se continuar a execução de um caso de teste suspenso. Durante a execução dos casos de testes manuais, tem-se a opção de suspender ou parar o teste.
- **Critério 4**: A ferramenta *TestManager* possui suporte a esse critério. Para executar um caso de teste específico ou uma *suíte* de teste com vários testes, tem-se essa opção para a escolha.

4.4 Acompanhamento dos Resultados da Execução dos Testes

A Tabela 5 mostra o resultado da análise desse critério nas ferramentas selecionadas.

Tabela 5: Acompanhamento dos Resultados da Execução dos Testes

			FERRAMENTAS		
ATIVIDADES	**ID**	**CRITÉRIOS**	***TESTLINK***	***TESTOPIA***	***TESTMANAGER***
Acompanhamento dos Resultados da Execução dos Testes	1	Acompanhar a execução dos teste em tempo real	Sim	Não	Sim

A seguir é dada uma definição dos critérios utilizados para analisar a atividade Acompanhamento dos Resultados da Execução dos Testes. Para o critério ser válido foram considerados os seguintes requisitos:

Critério 1: A ferramenta deve ter suporte a uma funcionalidade que acompanha a execução dos testes em tempo real, indicando ao usuário uma tela com as principais informações da execução.

4.4.1 FERRAMENTA *TESTLINK*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testlink* possui suporte a esse critério. Durante a execução dos casos de teste, o *Testlink* mostra na tela, quando o usuário executa o teste, as seguintes informações: data, *build*, testador, *status*, versão do caso de teste e anexos.

4.4.2 FERRAMENTA *TESTOPIA*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testopia* não possui suporte a esse critério.

4.4.3 FERRAMENTA *TESTMANAGER*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *TestManager* possui suporte a esse critério. Ela possui uma funcionalidade chamada *Monitoring Test Runs* onde, enquanto testa-se

manualmente, apresenta: qual testador está testando, quantos testes estão suspensos, terminados, parados, o tempo de duração do teste, entre outros.

4.5 Relatórios de Resultados de Teste

A Tabela 6 mostra o resultado da análise desse critério nas ferramentas selecionadas.

Tabela 6: Relatórios de Resultados de Teste

			FERRAMENTAS		
ATIVIDADES	**ID**	**CRITÉRIOS**	***TESTLINK***	***TESTOPIA***	***TESTMANAGER***
Relatórios de Resultados de Teste	1	Emitir relatórios	Sim	Sim	Sim
	2	Configurar o ambiente para gerar relatórios por papéis	Sim	Não	Não
	3	Gerar relatórios em diferentes extensões (Pdf, Word, Excel, HTML)	Sim	Não	Sim

A seguir é dada uma definição dos critérios utilizados para analisar a atividade Relatórios de Resultados de Testes. Para os critérios serem válidos foram considerados os seguintes requisitos:

- **Critério 1**: A ferramenta deve suportar a emissão de relatórios completos após a execução dos testes. Os relatórios devem conter informações sobre os casos de testes executados corretamente, com falha, e os que não foram executados. Também deve conter a informação sobre quem testou, entre outras informações.
- **Critério 2**: A ferramenta deve suportar a emissão de relatórios para diferentes tipos de usuários, como o líder de teste, convidado, administrador, testador, testador sênior. A ferramenta também deve ter a funcionalidade de permitir que o usuário logado no sistema, consiga visualizar o relatório de acordo com as permissões que lhe são concedidas.
- **Critério 3**: A ferramenta deve suportar a emissão de relatórios para diferentes tipos de extensões. Facilitando assim a visualização dos relatórios.

4.5.1 FERRAMENTA *TESTLINK*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testlink* possui suporte a esse critério. O *Testlink* contém relatórios completos sobre os planos de teste, métricas gerais do plano de teste, resultados por *build*, matriz de resultados de teste, casos de teste com falha, bloqueados, não executados, atribuídos a testadores, entre outros.
- **Critério 2**: A ferramenta *Testlink* possui suporte a esse critério. O *Testlink* é construído com seis diferentes níveis de permissões padrões. De acordo com o "papel", a gama de relatórios é maior ou menor. Por exemplo, o administrador consegue visualizar todos os tipos de relatórios. Já o testador, só consegue visualizar os que são permitidos a ele.
- **Critério 3**: A ferramenta *Testlink* possui suporte a esse critério. O *Testlink* consegue gerar relatórios em diferentes formatos, como: Word, Excel, HTML, *Open Office.*

4.5.2 FERRAMENTA *TESTOPIA*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *Testopia* possui suporte a esse critério. *Testopia* contém tabelas com o *status* dos casos de testes, a porcentagem dos testes realizados, as prioridades dos testes. Dado um intervalo de início e fim dos testes, ele gera uma tabela com quantos casos de teste foram executados nesse intervalo de tempo. Entretanto, *Testopia* gera somente tabelas com os resultados. Das três ferramentas analisadas, é o que mais deixa a desejar em relação aos relatórios de testes.
- **Critério 2**: A ferramenta *Testopia* não possui suporte a esse critério. O *Testopia* também possui diferentes níveis de permissão. Mas como ele é integrado com o *Bugzilla*, as permissões basicamente são relacionadas à criação e edição de *bugs*. O *Testopia* tem um "papel" de testador, exclusivo pro *Testopia*, onde o relatório é completo. Portanto, o *Testopia* possui um tipo de relatório.
- **Critério 3**: A ferramenta *Testopia* não possui suporte a esse critério. O *Testopia* gera relatório somente na ferramenta, ou seja, tem que estar logado no *Testopia*

para conseguir visualizar os relatórios, os quais são gerados somente em HTML. Não suportando outras extensões.

4.5.3 FERRAMENTA *TESTMANAGER*

Os resultados da avaliação dos critérios nessa ferramenta foram os seguintes:

- **Critério 1**: A ferramenta *TestManager* possui suporte a esse critério. Ela contém relatórios completos, sendo eles: uma lista de relatórios sobre configurações, *scripts*, planos de teste, usuários, *suítes* de teste, *logs*, entradas de teste, informações completas sobre a execução dos casos de teste, como porcentagem de testes executados, entre outros relatórios.
- **Critério 2**: A ferramenta *TestManager* não possui suporte a esse critério. Ela gera relatórios completos, mas sem distinção do "papel" do usuário, ou seja, os relatórios gerados contêm o mesmo conteúdo para todos os usuários e permissões.
- **Critério 3**: A ferramenta *TestManager* possui suporte a esse critério. O *TestManager* consegue gerar relatórios em diferentes formatos, como: Word, Excel, HTML, *Open Office*, PDF, CSV, XML.

4.6 Análise Geral

Com base nos resultados obtidos através da aplicação dos critérios de avaliação nas três ferramentas, tem-se como resultado geral um melhor desempenho analisado em duas ferramentas: *Testlink* e *TestManager*, onde a *Testlink* apresentou suporte para nove critérios e a *TestManager* suportou oito critérios dos dezoito propostos. Já a ferramenta *Testopia* apresentou sustentação em sete critérios.

Na análise dos critérios da atividade "Projetando Casos de Teste" notou-se uma maior eficiência nas ferramentas *Testlink* e *Testopia*. A ferramenta *TestManager* apresentou suporte apenas para os critérios de "Importar requisitos de diferentes extensões para os casos de teste" e "Organizar os casos de teste em *suítes* para um melhor gerenciamento".

Entretanto, na atividade "Construindo Casos de Teste" quem apresentou maior suporte foi a ferramenta *TestManager*. Sendo que, as ferramentas *Testlink* e *Testopia* não sustentaram nenhum dos critérios avaliados nessa atividade.

Observando-se a ação de "Executar Casos de Teste" notou-se uma melhor resposta das três ferramentas. Porém a ferramenta *Testopia* foi a que apresentou uma maior integração com

os critérios. Todavia no “Acompanhamento dos Resultados da Execução dos Testes” essa ferramenta não atendeu ao único critério proposto; já as ferramentas restantes atingiram o resultado esperado.

Enfim, na atividade de “Relatórios de Resultados de teste” houve uma melhor eficácia na ferramenta *Testlink*, que teve suporte para todos os critérios avaliados.

A *TestManager*, por ser uma ferramenta proprietária, não correspondeu às expectativas esperadas, pois apresentou uma igualdade no desempenho dos aspectos mais importantes de gestão de testes, juntamente com a *Testopia*, que é uma ferramenta *open source*. Salienta-se que esta análise limita-se aos critérios definidos por este trabalho. Porém, a *TestManager* destacou-se no critério “Acompanhar a execução de teste em tempo real” porque apresentou uma funcionalidade completa, com inúmeros recursos à disposição do testador (quantidade de testes suspensos, tempo de duração dos testes, porcentagem dos testes realizados).

Em contrapartida, a ferramenta *Testopia* apresentou relevante incapacidade no critério “Gerar relatórios em diferentes extensões”, pois gera apenas relatórios na própria ferramenta, em uma única extensão. Já no critério “Emitir relatórios” essa ferramenta apresenta suporte, mas com menor especificação e diversidade.

Por fim, a *Testlink* foi a ferramenta que expôs uma melhor eficácia na avaliação dos critérios. Destacou-se pela sua regularidade na sustentação de critérios relevantes na gestão de testes. Não apresenta opções de aperfeiçoamento para as funcionalidades, mas tem suporte para as ações necessárias.

5 CONSIDERAÇÕES FINAIS

Em decorrência das inúmeras ações complexas que envolvem o teste de *software*, surgem ferramentas de teste que auxiliam nesse processo de desenvolvimento. Este trabalho propõe um conjunto de critérios de avaliação para análise dessas ferramentas e realiza uma comparação envolvendo três ferramentas de gestão de teste com base nos critérios propostos, visando identificar a ferramenta que melhor suporta os critérios.

A escolha do processo tradicional de teste tornou-se uma alternativa eficaz para a aplicação dos critérios de avaliação, pois através dele pode-se realizar uma análise rápida e objetiva, referente ao suporte das atividades do processo de teste pelas ferramentas.

Outro aspecto importante dessa pesquisa é a comprovação da utilização de uma ferramenta conveniente e eficiente para as funcionalidades da gestão de teste. Com isso, tem-se um melhor aproveitamento nas etapas encontradas no processo de teste, resultando em uma otimização do tempo gasto nas ações de teste, diminuição dos riscos envolvidos nas atividades do processo de teste, sendo esse de suma importância para a garantia de qualidade do *software* desenvolvido.

Referente à análise dos resultados obtidos, pode-se identificar os pontos fortes e fracos de cada ferramenta. A ferramenta *Testlink* destaca-se pela sua capacidade em gerar relatórios completos e configurar o ambiente para gerar esses relatórios distintos para cada usuário; em contrapartida não é capaz de criar dependências entre os casos de teste. Porém, nota-se que seu suporte para as atividades necessárias para o processo de teste sobressaem-se na comparação com os pontos fracos dessa ferramenta.

Já a ferramenta *Testopia* apresenta como pontos fortes a criação de dependências de casos de teste e capacidade para reportar *bugs*; no entanto não acompanha a execução dos testes em tempo real e não suporta a geração de relatórios em diferentes extensões.

A edição dos *scripts* de teste (manuais) e a opção de escolha entre executar *suítes* de teste ou casos de teste separados, são os pontos de destaque da ferramenta *TestManager*. Ferramenta essa, que apresenta como pontos fracos a incapacidade na classificação dos casos de teste de acordo com a prioridade e não possuir um filtro de buscas para execução dos casos de teste.

Todavia, a avaliação proposta nessa pesquisa limitou-se apenas a três ferramentas. Trabalhos futuros incluem análise desses critérios de avaliação em ferramentas pertencentes a uma mesma classificação (*open source* ou proprietária) para uma comparação mais apropriada.

Contudo, o desenvolvimento de mais pesquisas nessa área, com o aprofundamento e aperfeiçoamento nos critérios de avaliação de ferramentas de teste, também são necessários para melhor comparação, análise e discussão dos resultados já existentes.

6 REFERÊNCIAS

BARBOSA, E. F.; MALDONADO, J. C.; VINCENZI, A. M. R. **Software Testing: Verification and Reliability.** 2. Ed., 20011.

CAETANO, C. **Automação e Gerenciamento de Testes: Aumentando a produtividade com as principais soluções Open Source e Gratuitas.** 1. ed., 2008.

CENTERS, T. E. **Centro de Avaliação de Ferramentas de Teste de Software.** 2009. Disponível em: <http://test-tools.technologyevaluation.com/pt/>. Acesso em: mai. 2011.

CORPORATION, R. S**. Mentor de Ferramentas - Realização de atividades de testes usando o Rational TestManager.** 2001. Disponível em: <http://www.wthreex.com/rup/toolment/testmgr/tm_tstmn.htm>. Acesso em: jun. 2011.

CRAIG, R.D.; JASKIEL, S. P. **Systematic Software Testing.** Boston: Artech House Publishers, 2002.

DELFIM, S. M. **Gestão de Defeitos.** 2008. Disponível em: <http://portalarquiteto.blogspot.com/2008/02/gesto-de-defeitos.html>. Acesso em: mai. 2011.

FREIRE, M. B.; NASCIMENTO, L. C.; RANGEL, D. **Testlink - Manual do Usuário.** 2008. Disponível em: <http://www.testexpert.com.br/?q=node/822>. Acesso em: mar. 2011.

GOSH, S.; MATHUR, A. P. **Interface mutation, Software testing: Verification and Reliability.** 4. ed., 2001.

GUIDE, T. T. **Classes of Test Tools.** 2006. Disponível em: <http://www.softwaretestinggenius.com/download/CBOK.pdf>. Acesso em: mai. 2011.

HENDRICKS, G. **Software Testing With Testopia.** 2010. Disponível em: <http://mozilla.org/projects/testopia>. Acesso em: abr. 2011.

IEEE, S. **IEEE Standard Glossary of Software Engineering Terminology – IEEE Std 610.12-1990.** New York, EUA, 1990.

ILLES, T.; HERRMANN, A.; PAECH, B. **Criteria for Software Testing Tool Evaluation - A Task Oriented View**. 2005. Disponível em: <http://www.springer.dmmk.de/fileadmin/_primium/downloads/publikationen/IHPR2005.pdf >. Acesso em: mai. 2011.

KOSCIANSKI, A.; SOARES, M. S. **Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** 2. ed. São Paulo: Novatec Editora, 2007.

MARCORATTI, J. C. **A Gestão de Requisitos.** 2006. Disponível em: <http://imasters.com.br/artigo/3860/des_de_software/a_gestao_de_requisitos/>. Acesso em: jun. 2011.

MOZZILA. **Testopia.** 2010. Disponível em: <http://www.mozilla.org/projects/testopia/>. Acesso em: mar. 2011.

MYERS, G. J. **The Art of Software Testing.** 2. ed. NeW Jersey: John Wiley, 2004.

NETO, A. C. **Qualidade de Software.** 2008. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=8035>. Acesso em: mar. 2011.

PRESSMAN. R. S. **Engenharia de Software.** 6. ed. São Paulo: McGraw-Hill, 2006.

ROCHA, A. R.; MALDONADO, J. C.; WEBER, K. C. et al. **Qualidade de software – Teoria e prática.** São Paulo: Prentice Hall, 2001.

SILVESTRI, E. **Word do TestManager.** 2003. Disponível em: <http://www.eduardosilvestri.com.br/SITE/ita/ce230/listas/listex3/index.asp>. Acesso em: jun. 2011.

SODRÉ, C. C. **Norma ISO/IEC 9126: Avaliação de Qualidade de Produtos de Software.** Londrina: Universidade Estadual de Londrina, 2006.

SOFTWAREQATEST, T. **Software QA and Testing Frequently.** 2011. Disponível em: <http://www.softwareqatest.com/qatfaq2.html#FAQ2_6b>. Acesso em: jun. 2011.

TEAMST**. Latest News - Testlink.** 2011. Disponível em: <http://www.teamst.org/>. Acesso em: mar. 2011.

WIKIPEDIA. **Teste de Regressão.** 2011. Disponível em: <http://pt.wikipedia.org/wiki/Teste_de_regress%C3%A3o>. Acesso em: mai. 2011.

Apêndice A – Casos de Teste desenvolvidos

<Caso de Teste 0> - <Teste de Fazer Login>:

Descrição: O usuário (aluno) deve ser capaz de fazer login no sistema (Portal do Aluno) e assim ter acesso às opções de discente do sistema.

Pré-condições: O sistema deve estar disponível e operante.

Pós-condições: Após o login ser bem-sucedido, o usuário deve ter acesso às opções de aluno do sistema.

Dados necessários: Os dados utilizados nesta operação são: a matrícula e senha do aluno.

<Caso de Teste 1> - <Teste de Matricular Disciplina>:

Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de se matricular em uma disciplina.

Pré-condições: A primeira condição é que o usuário deve estar logado no sistema. Outra condição importante é que esteja no período de matrícula, definido no calendário da UFSM.

Pós-condições: A primeira condição é que deve aparecer uma lista com todas as disciplinas aptas para aquele usuário cursar e seus respectivos horários. O usuário pode então escolher uma ou várias disciplinas da lista. Após a confirmação, as disciplinas deverão estar presentes no histórico e no horário do respectivo aluno.

Dados necessários: Os dados utilizados nesta operação são: a matrícula e senha do aluno, disciplina, turma e horário da disciplina e grade curricular do respectivo curso do usuário.

<Caso de Teste 2> - <Teste de Consultar Horário>:

Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de consultar o seu horário.

Pré-condições: O usuário deve estar logado no sistema.

Pós-condições: A opção de consultar horário deve estar habilitada. Após a seleção da opção deverá aparecer na tela o horário do respectivo aluno. O usuário ainda terá a opção de imprimir seu horário, caso queira.

Dados necessários: Os dados utilizados nesta operação são: a matrícula, senha e horário do aluno.

<Caso de Teste 3> - <Teste de Imprimir Horário>:

Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de imprimir o seu horário.
Pré-condições: O usuário deve estar logado no sistema e ter realizado a operação de consultar horário (estar na tela de consulta ao horário).
Pós-condições: A opção de imprimir horário deve estar habilitada na tela de consulta ao horário.
Dados necessários: Os dados utilizados nesta operação são: a matrícula, senha e horário do aluno.

<Caso de Teste 4> - <Teste de Consultar Histórico>:
Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de consultar o seu histórico.
Pré-condições: O usuário deve estar logado no sistema.
Pós-condições: A opção de consultar histórico deve estar habilitada. Após a seleção da opção deverá aparecer na tela o histórico do respectivo aluno. O usuário ainda terá a opção de imprimir seu histórico, caso queira.
Dados necessários: Os dados utilizados nesta operação são: a matrícula, senha e histórico do discente.

<Caso de Teste 5> - <Teste de Imprimir Histórico>:
Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de imprimir o seu histórico.
Pré-condições: O usuário deve estar logado no sistema e ter realizado a operação de consultar histórico (estar na tela de consulta ao histórico).
Pós-condições: A opção de imprimir histórico deve estar habilitada na tela de consulta ao histórico.
Dados necessários: Os dados utilizados nesta operação são: a matrícula, senha e histórico do aluno.

<Caso de Teste 6> - <Teste de Mudar Senha>:
Descrição: O usuário (aluno) deve fazer login no sistema e assim ter a opção de mudar sua senha de acesso ao sistema.
Pré-condições: O usuário deve estar logado no sistema.
Pós-condições: Depois de realizada a operação de mudar senha, a senha do usuário no banco de dados será alterada e o usuário só poderá fazer login com a nova senha de acesso.

Dados necessários: Os dados utilizados nesta operação são: a matrícula, senhas antiga e nova do aluno.