UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERNANDO BETTIOL LOPES

DESENVOLVIMENTO DE UMA METODOLOGIA APLICADA AO GERENCIAMENTO E ACOMPANHAMENTO DE TESTE DE SOFTWARE VIA WEB

CRICIÚMA, JULHO DE 2009

FERNANDO BETTIOL LOPES

DESENVOLVIMENTO DE UMA METODOLOGIA APLICADA AO GERENCIAMENTO E ACOMPANHAMENTO DE TESTE DE SOFTWARE VIA WEB

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel Ciência da Computação da Universidade do Extremo Sul Catarinense.

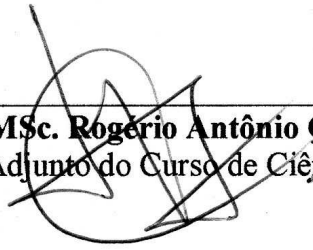
Orientadora: Prof MSc. Gustavo Bisognin

CRICIÚMA, JULHO DE 2009

FERNANDO BETTIOL LOPES

Desenvolvimento de uma Metodologia Aplicada ao Gerenciamento e Acompanhamento de Teste de Software Via Web

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

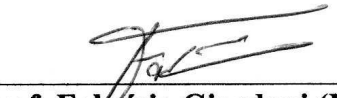


Prof. MSc. Rogério Antônio Casagrande
Coordenador Adjunto do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Gustavo Bisognin (UNESC)
Orientador

Profa. MSc. Ana Cláudia Garcia Barbosa (UNESC)



Prof. Fabrício Giordani (UNESC)

Aos meus queridos pais e meu irmão que sempre me apoiaram.

AGRADECIMENTOS

Primeiramente a Deus, que me abençoou durante todo
o tempo que estive cursando a faculdade, a minha família que sempre
me ajudou e incentivou nos momentos difíceis.
Agradeço também ao meu orientador Gustavo Bisognin por me orientar
e direcionar os caminhos certos para que pudesse desenvolver
o trabalho de conclusão de curso, agradeço a todos os professores por terem
passado seus conhecimentos, e também a todos
os colegas que sempre estiveram juntos comigo me apoiando
e incentivando nos momentos difíceis.

“No meio da dificuldade encontra-se a oportunidade.”

(Albert Einstein)

RESUMO

Este estudo propõe uma metodologia para auxiliar o processo e controle de testes em produtos de software. A Engenharia de Software é uma ciência que estuda de forma abrangente as etapas contidas na elaboração, desenvolvimento e manutenção de um sistema, sendo uma dessas etapas a fase de teste. Os testes em sistemas garantem a qualidade dos programas, evitando futuras falhas e perdas. Existem algumas técnicas e tipos de testes para facilitar e padronizar o trabalho do testador ou engenheiro de teste, as técnicas formam um conjunto de tipos de testes, onde o usuário que realiza a verificação possui a condição de escolher o tipo de investigação a executar em um determinado sistema. A metodologia proposta suporta a técnica de Caixa Preta, sendo um dos principais conceitos abordados neste trabalho. Como resultado da metodologia, foi desenvolvida uma ferramenta de apoio ao gerenciamento e de acompanhamento de testes de software que é executada em ambiente de rede e internet, permitindo independência de plataforma. Outro benefício da aplicação desenvolvida é o fato de permitir integração com ferramentas de controle de erros e acompanhamento de defeitos, além de fornecer uma metodologia de auxilio para o usuário testador de software.

Palavras-chave: Metodologia de testes, Engenharia de software, Técnicas de Testes, Tipos de Testes.

ABSTRACT

This study proposes a methodology to aid the process and control of tests in software products. The Engineering of Software is a science that studies in an including way the stages contained in the elaboration, development and maintenance of a system, being one of those stages the test phase. Test them in systems they guarantee the quality of the programs, avoiding future flaws and losses. Some exist techniques and types of tests to facilitate and to standardize the work of the tester or test engineer, the techniques form a group of types of tests, where the user that accomplishes the verification possesses the condition of choosing the investigation type to execute in a certain system. The proposed methodology supports Black Box's technique, being one of the main concepts approaching in this work. As a result of the methodology, a support tool was developed to the administration and of accompaniment of software tests that is executed in net atmosphere and internet, allowing platform independence. Another benefit of the developed application is the fact of allowing integration with tools of control of mistakes and accompaniment of defects, besides supplying a methodology of I aid for the software tester.

Word key: Methodology of test, Engineering of Software, Techniques of tests, Types of Tests.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Atual Mercado de software demanda cada vez mais por produtos de qualidade. Várias organizações estão exigindo o desenvolvimento de sistemas em um curto espaço de tempo. Estes sistemas necessitam ser escalonáveis e integrados com outros sistemas existentes ou em desenvolvimento. Outro fator importante, é que os ambientes tecnológicos onde estes sistemas são desenvolvidos estão em constante evolução.

A sociedade atual apresenta um crescimento significativo da implantação de sistemas, o que configura a grande dependência de boa parte dos negócios aos sistemas de informação. Neste novo contexto, passa a ser de vital importância contar com software de qualidade – softwares que fornecem resultados validáveis quando alimentados com dados válidos e que identificam corretamente entradas inválidas.

As disciplinas de verificação e validação de software atualizam-se diretamente conforme o crescimento da implantação do processo de teste pelas organizações desenvolvedoras de soluções baseadas e software. Ainda que as técnicas de teste de software mais utilizadas foram criadas por volta dos anos 70, as empresas têm uma grande dificuldade com a atividade de teste. Isto pode ser um reflexo da falta de profissionais especializados na área de teste de software ou mesmo da dificuldade em implantar um processo de teste utilizando as técnicas existentes na literatura (CRESPO et al, 2002).

Existem várias técnicas no mercado que são aplicadas no ciclo de vida de teste, sendo que uma das mais utilizadas é denominada de Caixa Preta (CRESPO et al, 2002).

A técnica denominada Caixa Preta trata da parte funcional de um software, trabalhando de maneira oposta ao teste de Caixa Branca, pois possibilita ao Engenheiro de Software dividir o sistema que irá testar em vários conjuntos, proporcionando assim que ele realize a verificação por partes ao invés de um todo (PRESSMAN, 2006).

Ela é responsável por estudar o comportamento do software que será testado, funcionando como um simulador, ela utiliza uma entrada de dados que seria aplicada ao software, assim verificando todos os seus processos para observar a sua funcionalidade. A técnica de caixa preta não testa a implementação de um software, mas sim a sua capacidade de executar as funções nele empregadas (SOMMERVILLE, 2003).

O teste é realizado de forma que uma entrada de dados seja aplicada, passando por uma verificação e no final comparam as saídas de dados, observando se elas possuem uma correspondência com aquela pré-definida pelo programador ou Engenheiro de Teste. Quando ocorre a correspondência entre as duas, então considera-se que o software testado não possui erros funcionais, caso contrário, observando-se variações entre as saídas comparadas, a ocorrência de erros foi detectada (SOMMERVILLE, 2003).

Na realização dos testes de software são empregadas situações específicas para cada técnica, tendo em vista que a técnica de Caixa Preta se preocupa em testar o software com visão de usuário. Assim, procurando uma situação onde o usuário tenha a possibilidade de utilizar o software com segurança de seu funcionamento, tendo também a certeza que as suas ações realmente estão sendo executadas pelo programa, sem ter a preocupação de saber como o software que está sendo utilizado foi criado, tanto na parte de implementação ou mesmo em sua linguagem de programação (KOSCIANSKI ; SOARES, 2006).

Observa-se que existem ainda hoje, muitos sistemas que apresentam problemas na execução de suas funções, onde os usuários muitas vezes demoram para entender e aplicar a ação desejada no software.

Mediante isto, esta pesquisa propõe o desenvolvimento de uma ferramenta que apóia o gerenciamento e acompanhamento de testes de software em sistemas que são produzidos pelos desenvolvedores ou Engenheiros de Software. A utilização da metodologia proposta apóia completamente o método de caixa preta, possibilitando o cadastramento de

casos de teste baseados na especificação dos requisitos dos sistemas a serem validados, bem como o acompanhamento da execução dos testes e gerenciamento dos defeitos (issue) encontrados durante a execução.

1.1 OBJETIVO GERAL

Desenvolver uma ferramenta para gerenciamento e controle efetivo de teste de software abordando os conceitos definidos pela técnica de caixa preta.

1.2 OBJETIVOS ESPECÍFICOS

Compreender Engenharia de Software;

Entender os testes de software por meio da técnica de Caixa Preta;

Aplicar os conhecimentos da técnica de Caixa Preta na criação de uma ferramenta de gerenciamento de testes de software;

Acompanhar a vida dos possíveis erros nas funcionalidades de um software até o seu fechamento;

Proporcionar o gerenciamento e acompanhamento efetivo de testes de software por meio de uma ferramenta automática.

1.3 JUSTIFICATIVA

A crescente evolução dos computadores na ultima década proporcionou um aumento considerável no desenvolvimento de sistemas computacionais acarretando em um grande numero de sistemas mal especificados e defeituosos.

A criação de novos softwares requer alguns cuidados, visto que sistemas que apresentam funcionalidades duvidosas podem acarretar graves transtornos para o usuário e imediatamente para o seu criador. Para a diminuição da possibilidade de ocorrência de erros, os programas passam pela etapa de teste.

A etapa de teste verifica a ocorrência de prováveis falhas na execução do sistema. Esta fase do ciclo de vida de um software possui algumas técnicas, e dentro destas, existem tipos de testes que podem ser utilizados no momento da verificação dos erros.

Os testes são utilizados dependendo da necessidade de verificação de cada sistema, porque cada um possui uma função específica, sendo que um teste de funcionalidade pode ser realizado juntamente com o teste de usabilidade otimizando os casos de teste para estas metodologias.

O aumento da complexidade dos sistemas atuais exige uma validação criteriosa das funcionalidades dos produtos de software. A execução de testes que na sua maioria eram realizados manualmente tornou-se inviável, pois a grande complexidade de funcionalidades existentes exige uma abordagem automatizada ou semi automatizada da execução destes casos de teste. A realidade atual compõe uma série de ferramentas e técnicas diretamente voltadas a validação e verificação de modelos de software, contudo o papel do testador e analista de teste ainda é extremamente importante para o processo de qualidade em uma linha de produção de sistemas.

Uma maneira encontrada para a melhoria desta condição foi à elaboração de ferramentas que detectam erros de forma automática, onde elas verificam o sistema e avaliam a sua integridade.

A utilização desses dispositivos auxilia o Engenheiro de Teste em suas atividades, mas houve a necessidade de uma ligação entre o sistema a ser testado e a ferramenta testadora, desta forma ocorreu o desenvolvimento de um programa que utiliza uma

metodologia de aplicação permitindo a integração entre os dois sistemas.

A metodologia possibilita ao testador cadastrar todas as informações necessárias para o desenvolvimento e execução do seu plano de teste. A ferramenta permite também que sejam cadastrados os usuários que fazem parte do grupo de teste, informando os seus papéis dentro desse grupo.

A utilização da ferramenta com essa metodologia de aplicação disponibiliza algumas vantagens como:

a) gerenciamento total de cadastros sejam eles de perfis, usuários ou testes;

b) permite a inserção das informações necessárias para a criação de um plano de teste;

c) execução do plano de teste integrando esse plano a uma ferramenta de verificação;

d) exibição de relatórios com informações pertinentes sobre a verificação.

O melhor desempenho durante o teste de software acarreta para o engenheiro algumas vantagens:

a) uma manutenção mais rápida impedindo que esse erro cresça tornando-se uma situação onde o engenheiro ou desenvolvedor tenha que abandonar o seu software e recomeçar um novo;

b) a entrega de seu sistema no prazo determinado, pois se conseguir resolver o problema dos erros antes do término do sistema, muito provavelmente o software poderá ser implantado para o usuário na data predefinida;

Os sistemas precisam ser testados minuciosamente para que sejam entregues com a menor quantidade de falhas possíveis. Com base nesta premissa, este trabalho busca apoiar a qualificação dos sistemas computacionais em uma linha de desenvolvimento de software.

2 ENGENHARIA DE SOFTWARE

A Engenharia de Software possibilita o controle sobre os processos de desenvolvimento de software pelo engenheiro, assim oferecendo a oportunidade de criação de sistemas de alta qualidade, fornecendo as bases nas quais os programas devem ser desenvolvidos, a área que atua na criação de software possui três etapas indispensáveis: métodos, ferramentas e procedimentos (INTHURN, 2001).

Os métodos oferecem ao desenvolvedor um caminho para que eles tenham condições de seguir na sua criação, ele utiliza-se de várias tarefas, onde é possível encontrar: o planejamento, uma analise apurada sobre os softwares, estudo da estrutura de dados, projeto de sistema de processamento, codificação, manutenção de sistema e testes (SOMMERVILLE, 2003).

As ferramentas buscam auxiliam os métodos utilizando apoio automatizado ou até mesmo semi-automatizado. Há ocorrência de ferramentas que possibilitam esses suportes aos métodos separadamente, mas quando elas são unidas de maneira que o resultado de uma possa ser utilizado em outra, então alcançam o seu objetivo principal, um sistema de suporte para a criação de software (PETERS; PEDRYCZ, 2001).

Os procedimentos formam uma ligação entre os métodos e as ferramentas, colaborando na criação racional de programas de computadores. Eles aplicam a forma em que os métodos serão utilizados, são também controles que tem a função de garantir a qualidade e controlar as mudanças (SOMMERVILLE, 2003).

A Engenharia de Software possui as três etapas citadas anteriormente, existe outra denominação para as etapas, que é o paradigma de Engenharia de Software. A escolha de um paradigma é baseada no objetivo que se deseja alcançar dentro de uma aplicação, levando em

conta os métodos e as ferramentas que serão utilizados, também observando os controles que se farão necessários (PRESSMAN, 2006).

2.1 CICLOS DE VIDA DE UM SOFTWARE

A Engenharia de Software além de estudar etapas e paradigmas que surgem em meio aos softwares, ela também volta as suas atenções para o modo de criação desses softwares. A intenção de realizar esses estudos voltados à criação de novos sistemas é de criar padrões que de alguma forma se eleve o nível do software criado para atingir uma melhor execução do mesmo, para o desenvolvimento de sistemas, a Engenharia de Software criou etapas que podem ser seguidas pelos desenvolvedores durante a criação de seus sistemas (INTHURN, 2001). Estas etapas podem ser descritas como:

a) estudo inicial: essa é a etapa inicial do projeto, onde nela são feitas as entrevistas para identificar as necessidades que o usuário procura resolver, terminando essa parte então se inicia o projeto propriamente dito, com as funções que serão utilizadas, os recursos necessários, os custo entre outros;
b) análise: a fase de analise trabalha as informações que foram adquiridas na fase inicial transformando-as em funções que o sistema utilizará, essa transformação é feita pelo método de criação de diagramas, sendo na etapa de analise que são utilizados os Diagramas de Fluxo de Dados e Entidade Relacionamento, mas todos esses tipos de diagramas são realizados na Analise Estruturada, pois na Analise Orientada a Objeto é utilizado diagrama de caso de uso e diagrama de classe ambas utilizando uma ferramenta de modelagem Unified Modeline Languagem (UML);

c) projeto: é na elaboração do projeto que são divididas as tarefas para os membros da equipe, observando que dentro das tarefas estão as atividades que possuem a função de oferecer uma seqüência aos módulos que estão sendo implementados, fazendo com que eles sigam a seqüência predefinida na etapa anterior. A fase do projeto também tem a característica de relatar os recursos de hardware, como rede, banco de dados, capacidade do servidor;

d) implementação: a implementação trata do desenvolvimento do sistema utilizando os requisitos obtidos nas fases anteriores, nesta fase ocorre a transformação das necessidades descrita pelo o usuário na fase inicial em código de programa, ainda nesta etapa é feito um teste para descartar os erros de programação e também os erros de funcionalidade;

e) testes: após a implementação é necessário então começar os teste para corrigir certos problemas que poderiam ter ocorridos, como na fase de implementação os módulos foram criados separadamente, nos testes não é diferente pois esses módulos são testados um a um e após um teste geral, com o sistema inteiro. A realização dos testes é feito de forma onde é feita uma pré-definição de erros e como eles devem ser descritos e resolvidos, assim observando os resultados obtidos com os resultados esperados;

f) documentação: na etapa da documentação são criadas todas as documentações do sistema, como seus manuais tanto de instalação como o de usuário, demonstrando de forma clara e detalhada como o usuário deve trabalhar com o sistema;

g) instalação: é basicamente a entrega do manual e do sistema final, com todas as suas funcionalidades e operando de forma eficiente e sem erros provenientes.

O processo de desenvolvimento bem elaborado acarreta uma boa qualidade de software. Para a realização de uma estrutura de desenvolvimento satisfatória pode-se seguir as etapas de criação de software conforme a Figura 1 (INTHURN, 2001):

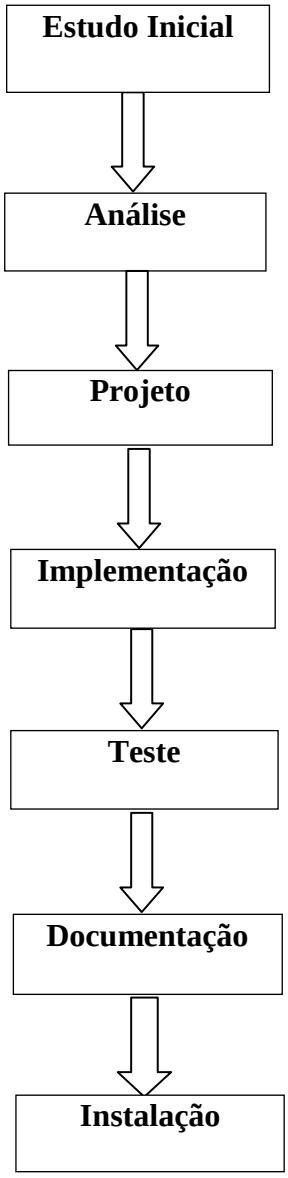


Figura 1. Ciclo de desenvolvimento de software
Fonte: INTHURN, C.(2001).

Um desenvolvimento de software bem sucedido ocorre com a utilização das etapas citadas anteriormente, pois é nesta fase que ocorre a definição do ciclo de vida de um sistema, baseando-se na boa ou má utilização das etapas descritas na fase de criação.

O ciclo de vida Figura 2 de um software possui variações tanto para um ciclo mais longo ou um ciclo mais curto, para que o sistema possua um tempo de vida mais longo é necessária uma boa estruturação das fases que estão presentes em seu ciclo de desenvolvimento (INTHURN, 2001).

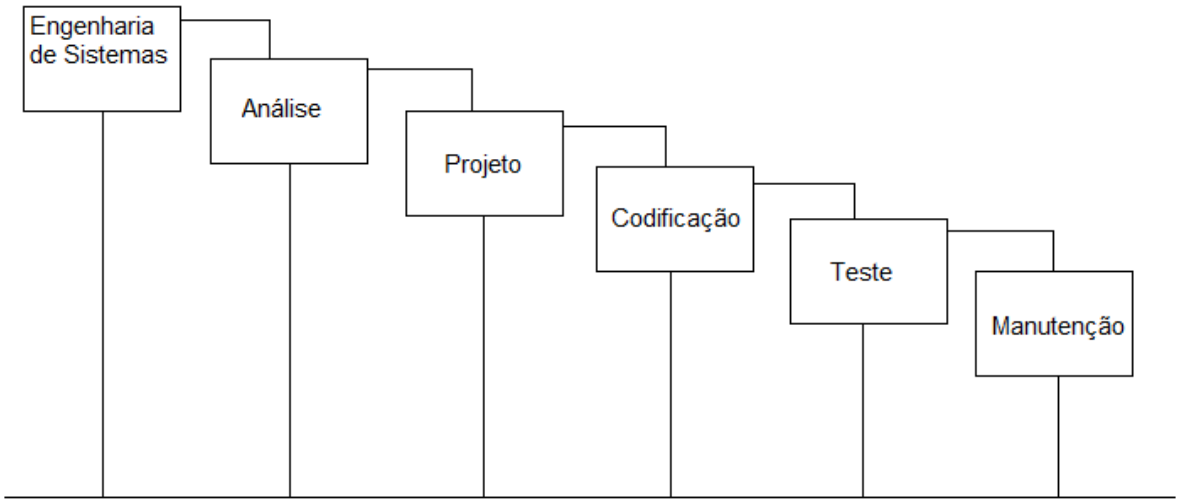


Figura 2. Ciclo de vida de um software
Fonte: ROCHA, C; ROCHA, T; ALMEIDA, T (2003).

A utilização de todas essas etapas na criação de um software se faz necessário, mesmo assim há possibilidade da ocorrência de erros dentro do programa. Esses erros podem ser conseqüência de vários fatores, como durante a implementação ou até mesmo erros de funcionalidade.

A funcionalidade não trata de sintaxe ou erro de código, mas sim das funções existentes no programa, avaliando se elas estão executando as operações nas quais foram definidas, assim proporcionando ao usuário uma confiança que suas ações estão sendo executadas de forma correta pelo programa, afastando os perigos de ocorrer prejuízos para o usuário.

A busca desses erros é efetuada na forma de testes, os softwares depois de prontos ou mesmo durante o seu desenvolvimento passam por diversos tipos de testes para chegar em uma qualidade satisfatória para o usuário.

3 TESTE DE SOFTWARE

O teste de software requer aplicações de várias atividades sobre o programa, onde se espera no final uma resposta em que o desenvolvedor tenha a possibilidade de observar se o seu programa realiza as funções nas quais ele foi inicialmente projetado, observando também que ele continuará executando as suas funções mesmo para dados diferentes a aqueles utilizados no teste (GUSTAFSON, 2002).

A realização dos testes se faz importante, pois por mais que se tente criar softwares perfeitos, eles provavelmente possuíram algum tipo de erro. A existência de erros ocorre por algumas razões, seja ela por desenvolver um programa para um cliente que ainda não está certo do que realmente seu sistema necessita, também podem aparecer defeitos quando se tem um projeto grande com um numero de pessoas envolvidas elevado, observando muitas idéias e na hora da implementação pode ocorrer uma divergência entre as funções.

Quando se ouve a expressão o sistema falhou, significa de maneira geral que ele não executou de forma correta as funções nas quais foram definidas e que são as mesmas que o usuário apontou como suas necessidades. Observando por esses ângulos pode-se dizer que a falha ocorre devido ao mau funcionamento de um ou mais aspectos do sistema (PFLEEGER, 2004).

Os testes de software possuem níveis que devem ser respeitados na hora de executar o procedimento de validação, onde é preciso que ocorra uma verificação funcional e estrutural de seus componentes. Os quatros níveis de teste de software podem ser descritos por: (PETERS; PEDRYCS, 2001)

a) componente é a parte onde acontece uma verificação sobre os elementos que fazem parte de um projeto e que foram implementados, compõem esses elementos as funções e os módulos. Esse nível possui a função de fazer com

que a lógica funcione de forma correta, e comprovar que os componentes estão realmente funcionando de forma como foram projetados;

b) integração trabalha na junção dos elementos físicos e lógicos do sistema, assim procura realizar a integração do software com o hardware que o programa ira precisar para a sua execução, onde a integração é executada até que os dois componentes estejam funcionando de forma linear e correspondente a aquilo no qual foi projetado. A integração possui a função que presta a garantia de satisfação dos objetivos que foram designados no projeto realizado para o sistema;

c) sistema é a globalização de tudo que é necessário para a sua execução, desde o hardware ate o software. Exercendo a função de unir esses dois componentes de forma homogênea, assim produzindo um melhor desempenho do software, ele ainda testa a funcionalidade do programa, para mostrar que ela está funcionando de forma correta e igual aquela na qual foi designada;

d) aceitação verifica os resultados finais dos testes e então pode determinar se o que foi testado está dentro das expectativas qual foram feitas durante a criação do projeto de sistema, possuindo a função de estabelecer uma satisfação dos participantes dos projetos perante os objetivos alcançados durante a execução do teste.

Os testes de software além de possuir seus níveis para que eles sejam realizados, devem também possuir um objetivo do seu acontecimento, pois os sistemas não são testados somente para dizer aos usuários que irão adquiri-los que são bonitos, mas sim mostrar que possuem a função que se espera dele.

3.1 OBJETIVO PARA TESTE DE SOFTWARE

Existem regras para a execução dos testes, que buscam padronizar, agilizar e servir aos engenheiros de software, sendo de fundamental importância observar os objetivos da atividade de teste, entre os objetivos dos testes pode-se citar alguns (PRESSMAN, 2006):

a) o teste é executar um sistema com o objetivo de encontrar erro.
b) um caso de teste de boa qualidade é aquele que possui um alta possibilidade de descobrir um erro que ainda não tinha se observado.
c) o teste bem feito é aquele que consegue encontra um erro que ainda não tinha sido encontrado.

Os objetivos citados anteriormente demonstram uma quebra de paradigma, pois eles dizem que um teste para ser bem sucedido deve encontrar erros, principalmente erros que ainda estavam ocultos, por esse motivo pode-se dizer que um teste apenas por não ter encontrado nenhum erro foi um teste bem sucedido. A meta é alcançar erros de diferentes tipos, fazendo isso em um curto espaço de tempo. Contudo se houver um sucesso na realização da atividade de teste observando que foram seguidos todos objetivos estabelecidos, então ela certamente encontrará erros no sistema. A vantagem que alem de descobrir o erro, junto a isso ela consegue mostrar que as funções definidas estão funcionando de maneira correta (PRESSMAN, 2006).

3.2 TIPOS DE ERROS

A pesquisa de erros muitas vezes é feita após o termino de toda a implementação e compilação dos programas, com a finalidade de buscar por erros que não são visíveis ou mesmo por aqueles que de alguma maneira ainda não foram detectados, mas para que haja a

possibilidade de encontrar e corrigir esses erros é necessário que se saiba que tipo de problema está sendo procurado (PFLEEGER, 2004).

A ocorrência de um erro acontece em um programa quando uma lógica de seu componente não possui uma saída adequada para uma entrada definida, isso aparece geralmente porque alguma etapa do processo do sistema está errada. Os erros algumas vezes são fáceis de serem localizados, basta apenas a realização de um teste de mesa simples, existem alguns tipos de erros mais freqüentes nos programas, que são eles (PFLEEGER, 2004):

a) teste da condição errada;
b) esquecer de inicializar as variáveis ou definir invariantes do loop;
c) esquecer de testar uma condição em particular;
d) comparar variáveis de tipos de dados inadequados.

Os erros causam perdas nos cronogramas de desenvolvimento de um engenheiro, pois ao compilar o sistema e observar que o mesmo possui um ou mais erros, o desenvolvedor terá a função de voltar e rever aquilo que foi criado até o momento, sem poder continuar em seu trabalho, assim atrasando o desenvolvimento do sistema.

3.3 TIPOS DE TESTES

Os tipos de testes ajudam o engenheiro na hora de estabelecer um plano de teste, pois é com eles que o engenheiro ira determinar e distribuir as tarefas que serão utilizadas na composição da realização dos testes (INTHURN, 2001).

A existência de uma grande quantidade de tipos de testes auxilia os engenheiros nas diversas formas de testar um sistema, a utilização de varias técnicas durante a execução de um teste é importante, pois nenhuma técnica deve trabalhar sozinha, elas devem completar

umas as outras assim montando uma grande estrutura de realização de teste mais eficaz (INTHURN, 2001).

3.3.1 Verificação e validação

O teste de verificação e validação trabalha a pesquisa de erros no software, fazendo com que o software verificado esteja dentro das especificações que foram definidas durante a sua criação. A verificação e validação funcionam como um ciclo de vida, testando o sistema desde o seu início, quando é realizado o projeto para a sua criação (SOMMERVILLE, 2003).

O teste avalia a funcionalidade de um sistema observando se ele realmente cumpre o que está espeficicado nele, o teste funciona realizando a verificação se todos os requisitos estão executando conforme o proposto, a validação trabalha avaliando se o sistema que está sendo desenvolvido é aquele sugerido para que fosse desenvolvido.

3.3.2 Teste de Unidade

O teste de unidade busca a avaliação e verificação nas menores unidades pertencentes ao projeto de software elaborado, para a execução dos testes são utilizados os projetos de sistema como uma forma de orientação para que ocorra uma verificação de maior eficácia dentro dos módulos dos sistemas. A dificuldade de compreensão que ocorre nesse teste torná-lo uma forma de avaliação, onde tanto ele como as repostas obtidas de seu estudo limitadas, devido ao seu método de avaliação que é restrito, então o teste pode dividir o software e avaliar partes do sistema (PRESSMAN, 2006).

A realização da divisão auxilia o teste a avaliar cada situação do programa de forma eficaz e com uma maior precisão nas suas repostas obtidas devido as suas avaliações (INTHURN, 2001).

As Interfaces possuem a função de fazer com que as informações sejam inseridas de forma correta no sistema, recebendo dados e dando as repostas necessárias para o usuário, que no momento em questão é o Engenheiro de Software trabalhando sobre o sistema realizando teste (PRESSMAN, 2006);

a) estruturas de dados funcionam na expectativa de conservar os dados existentes dentro do software de forma que eles mantenham a sua integridade (INTHRUN, 2001);
b) condições limites como o próprio nome diz, leva o sistema a executar no seu limite, fazendo com que seus módulos funcionem mesmo em condições criticas para poder observar suas restrições (PRESSMAN, 2006);
c) caminhos independentes funcionam de forma onde se tenha a certeza que os caminhos do sistema foram aplicados pelo menos uma vez (INTHRUN, 2001).
d) caminho de tratamento de erros ele trata de observar a ocorrência de possíveis valores verdadeiros, levando em consideração se esses valores estão passando pelo tratamento correto durante o teste (PRESSMAN, 2006).

O teste de unidade pode ser utilizado também em sistema que foram criados com orientação a objeto, mas para esse tipo de sistema o teste se torna mais complexo, pois na orientação a objeto ocorrem muitas atividades que anteriormente se tinha em um sistema comum, possivelmente nesse tipo de software é possível encontrar: classe, encapsulamento, herança entre outro. A existência de tudo isso eleva a dificuldade do teste de unidade, pela razão de aumentar seu campo de atuação (INTHURN, 2001).

3.3.3 Teste de Integração

O teste de integração avalia os módulos que foram testados e executados um de cada vez, agora está desenvolvendo o mesmo resultado depois de integrar um com os outros, isso significa que ele tem a função de verificar se os módulos utilizados pelo sistema estão funcionando de forma correta quando estão juntos. O teste de integração utiliza a forma incremental para realizar as avaliações sobre o sistema, integrando os módulos que passaram pelo processo de avaliação no sistema completo (INTHURN, 2001).

O teste de integração pode possuir algumas fases que ajudam a desempenhar o seu papel como um avaliador, essas fases executaram uma avaliação de maneira funcional, avaliado a funcionalidade do sistema que passa pelo processo de teste, as fases pertencentes a este teste pode ser definidas como (PRESSMAN, 2006):

a) integração: é nessa parte do teste que são criadas todas as características do sistema, ou seja, aqui é montado toda a representação que será passada para o usuário, variando desde o funcionamento do sistema com sua interface, até os erros que o sistema apresentará, mas que será solucionado pelos engenheiros antes de sua entrega;
b) manipulação e análise de dados: nessa etapa são realizados os trabalho com os dados que o sistema apresentará, essa manipulação pode variar sendo utilizado de algumas maneiras que podem passar pela dimensão de símbolos e de dados e rotação dos mesmos;
c) processamento e geração de *displays* na fase descrita são criados os gráficos, as imagens e interfaces que o sistema usara durante a sua execução;
d) gerenciamento de banco de dados nessa fase ocorre às ações que o sistema exercerá sobre o banco de dados que estará utilizando, aqui é possível que o

testador avaliei as informações do banco de dados, como acesso, manipulação entre outros.

As fases que o teste de integração possui, exerce a função de realizar testes funcionais nos sistemas, mesmo que ocorra um relacionamento a um domino de estrutura de dados (PRESSMAN, 2006).

3.3.4 Teste de Validação

O teste de validação segue o teste de integração, dessa maneira alguns erros que provavelmente seriam passados para frente que estão resolvidos no teste anterior, esse teste possui a função de avaliar se o sistema está correspondendo de maneira correta, ou consideravelmente correta durante a execução do teste (PRESSMAN, 2006).

A validação de um sistema é obtida de maneira onde é feitos vários teste com o sistema, sendo que eles são teste que pertencem à técnica de Caixa Preta, dessa maneira tem a função de avaliar então a funcionalidade do sistema em questão. A validação também trata das atividades de configuração que o sistema possui, dessa forma avaliando os seus requisitos de configuração para observar se não foi esquecido de desenvolver nenhum, ou se eles estão trabalhando de maneira correta no momento que são solicitados pelos usuários (INTHURN, 2001).

3.3.5 Teste de Sistema

O teste de sistema possibilita não somente os engenheiros de software realizam testes de um sistema, o sistema é um conjunto de elemento como hardware e software. Devido à integração de várias partes em um único sistema, ocorre de aparecer erros em locais

que não foram criados pelos engenheiros que estão estando o sistema naquele momento. Contudo para evitar que isso ocorra, o engenheiro deve se preparar anteriormente para esses tipos de acontecimento, então realizar um tratamento para erros que parecem serem de maiores ocorrência dentro de um sistema (PRESSMAN, 2006).

O teste de sistema possui a característica de ser um teste de validação, pois ele é utilizado para mostrar que o conjunto (sistema integrado com todas as suas funções e atribuições) está funcionando de maneira correta. A sua característica de validação leva o teste para a área funcional de um sistema, alem de testar tudo o que envolve o sistema de software, ele também avalia o conjunto lógico junto com o hardware, para observar se o seu funcionamento atribuído ao software está de maneira regular e sem erros (INTHRUN, 2001).

A avaliação que acontece nesse teste, não ocorre isoladamente utilizando um único meio de avaliação, ele possui quatro etapas que juntas constituem o teste de sistema, as etapas são (PRESSMAN, 2006):

a) teste de recuperação: esse teste trabalha imprimindo uma avaliação onde leva os sistemas que estão passando pelo teste a realizarem falhas, ele faz isso para que os engenheiros tenham a possibilidade de observar se as falhas que aconteceram serão recuperadas de forma correta. A recuperação de um sistema pode ocorrer de duas maneiras, humana ou máquina, caso a recuperação seja automática ou da maquina ela deve reiniciar e realizar o teste novamente para cada falha encontrada, caso a recuperação seja humana então devem-se avaliar o tempo levado desde a sua localização da falha até a sua recuperação (INTHURN, 2001);
b) teste de segurança: os sistemas de computadores sendo privado ou mesmo de uso domestico possuem a probabilidade de serem invadidos de uma maneira ilegal, para evitar que acontece esse tipo de atitude, o teste de segurança avalia

a capacidade que o software possui de se manter mesmo durante a tentativa de uma invasão ilegal. Os testes dessa categoria devem ser feitos pelos engenheiros que não participaram ou criaram o sistema testado, pois os avaliadores testam o sistema tentando invadi-lo (PRESSMAN, 2006);

c) teste de estresse: esse tipo de teste possui a função de colocar o software no seu limite, para com isso determinar os seus pontos francos e suas limitações, essas tentativas de levar o sistema até o seu limite pode ser feitas de algumas maneiras como executar uma alta taxa de processamento para o software, ou uma grande quantidade de acesso ao banco de dados durante a aplicação do sistema (INTHURN, 2001);
d) teste de desempenho: possui a função de observar se o desempenho anteriormente definido para o programa realmente está ocorrendo durante a sua execução, para comprovar isso o teste de desempenho verifica o tempo de execução em varias partes do código, e após comparar com o tempo de resposta, assim comprovando o seu desempenho (PRESSMAN, 2006).

Os testes de software são importantes para mostrar aos desenvolvedores que existem erros em suas criações, sendo no início, meio ou final de seus projetos.

O software que passa pelo teste tem seus erros corrigidos, mas a alta quantidade de tipos de erros exige a necessidade de se possuir uma variedade de tipos de teste, sendo que cada tipo utilizado em uma situação diferente onde se deseja testar o sistema.

4 TÉCNICAS DE TESTES

As técnicas de testes ajudam a proporcionar ao engenheiro a indicação das limitações que cada tipo possui, dessa maneira pode se considerar necessários os vários tipos de teste de software existentes, sendo que cada um para uma determinada tarefa a ser realizada quando testado um programa (PETTERS; PEDRYCZ, 2001).

4.1 TÉCNICA DE CAIXA BRANCA

Técnica de Caixa Branca utiliza a forma de estrutura de controle para dividir os métodos e assim executar o teste desejado sobre o software, com esse tipo de teste o engenheiro tem a possibilidade de aplicar o teste de maneira e forma diferentes, entre essas maneiras estão (PRESSMAN, 2006):

a) a certeza que todas as possibilidades de funções existentes dentro de um modulo foram executadas pelo menos uma vez;
b) a execução de todas as aplicações lógicas existentes dentro do sistema, sendo elas falsas ou verdadeiras;
c) a execução dos laços pertencentes a um programa, isso acontece aplicando uma execução que levam esses laços a executarem no seu limite;
d) a garantia que as estruturas de dados que se encontram no interior do software sejam validas.

A execução do teste de Caixa Branca procura testar o que o sistema produz, executa. A verificação dessa atividade trabalha no detalhes do código fonte, ou seja, para realizar o teste de Caixa Branca sobre um sistema é necessário conhecer o seu código fonte (PETTERS; PEDRYCZ, 2001).

4.2 TÉCNICA DE CAIXA PRETA

O teste Funcional, esse tipo de teste é utilizado na execução do programa vistoriando a sua funcionalidade, de maneira onde são inseridas entradas de dados e observando a sua resposta após passar pelo software (INTHURN, 2001).

O teste funcional também é conhecido pelo teste de Caixa Preta, pois partem do pressuposto que a avaliação ocorrerá sobre o código de um sistema, mas essa avaliação não fará vistoria se o código está implementado de maneira correta, sem erro de sintaxe e sim observar se as funções presentes no sistema executaram todas as atividades nelas estabelecidas (PETERS; PEDRYCZ 2001).

O teste é realizado de maneira onde são inseridas entradas de dados, onde depois de realizado todo o teste sobre o sistema é observado às saídas, então essas saídas que foram obtidas como resultado dos testes é comparada com as saídas que estavam predefinidas pelos engenheiros, assim comparando se forem idênticas ou não, caso a reposta seja não então foi encontrado um erro no programa (SOMMERVILLE, 2003).

O teste de Caixa Preta possibilita a divisão de atividades dos testes em partes, assim auxiliando os engenheiros no momento de realização do mesmo. Essa divisão pode ocorrer desde a interface ate mesmo ao banco de dados que o sistema esta utilizando (PRESSMAN, 2006).

4.3 TÉCNICA ESTRUTURAL

A técnica do teste de Estrutura é utilizada em programas pequenos, ou pequenas rotinas, buscando verificar a estrutura de dados que o sistema está implementado, assim ele trabalha sobre o código fonte de um sistema (SOMMERVILLE, 2003).

O teste Estrutural busca tratar da lógica interna de um sistema, para esse procedimento podem ser utilizadas categorias que estão implantadas dentro da técnica (PRESSMAN, 2006).

A categoria de teste conhecida como abrangência de instrução, necessita que todas as declarações existentes dentro do programa sejam executadas pelo menos uma vez, dessa forma ela é considerada uma categoria fraca para a realização de testes (SOMMERVILLE, 2001).

A ramificação, nessa categoria todas as ramificações existentes nos sistemas são executadas pelo menos uma vez, para a realização desse teste é necessário um funcionamento de verdadeiro ou falso nos blocos de seleção feitos pela categoria de teste (PETERS; PEDRYCZ, 2001).

O caminho trata da execução das atividades dentro do caminho que o sistema possui, dessa maneira o sistema necessita que cada caminho seja executado pelo menos uma vez, para que assim essa categoria de teste consiga realizar os testes solicitados pelo engenheiro (PETERS; PEDRYCZ, 2001).

4.4 TÉCNICA CAMINHO BÁSICO

A técnica do Teste de Caminho tende a possuir as mesmas características que o teste de estrutura, mas aqui o objetivo é fazer com que se execute pelos caminhos para que o teste seja realizado. Observando-se que cada caminho possui declarações em seus componentes, dessa forma se pode deduzir que essas declarações também foram executadas também pelo menos uma vez. As declarações são testadas normalmente no caso de ocorrer um verdadeiro ou falso (SOMMERVILLE, 2003).

O caminho básico possibilita ao engenheiro que execute uma derivação de método, isso é feito para felicitar a compreensão do projeto de sistema que possuam uma complexidade alta, após a criação dessa derivação o engenheiro pode utilizá-la para facilitar a criação de um conjunto de caminho dentro do projeto, isso faz com que o desenvolvedor tenha a garantia de que utilizando essa derivação às instruções definidas no sistema irão executar pelo menos uma vez.

5 A TÉCNICA DE CAIXA PRETA

A técnica de Caixa Preta também conhecida como testes de funcionalidade, busca tratar da parte operacional de um sistema, procurando observar se há uma obediência na execução daquilo predeterminado na elaboração do sistema (KOSCIANSKI; SOARES, 2006).

A Caixa Preta tendo a função de avaliar a execução operacional do software e funcionando como uma “Caixa Preta” isolada torna-se improvável o estudo de suas ações durante a execução do teste, então apenas é possível observar as suas variações no termino do teste. Assim o Engenheiro de Software elabora seus estudos utilizando a entrada de dados no sistema criado, e observando as suas saídas (SOMMERVILLE, 2003).

O teste citado funciona de maneira contraria ao teste de Caixa Branca, o Caixa Branca é realizado no princípio da criação de software, enquanto o teste de Caixa Preta é realizado no período final de elaboração de um sistema (PRESSMAN, 2006).

A funcionalidade de um sistema não trata de sua implementação, mas sim de como ele se comportará durante a entrada de dados e a execução dos mesmos, existem algumas categorias para que se possa realizar o teste de funcionalidade, como (PRESSMAN, 2006):

a) a presença de funções que se apresentam de forma incorreta;
b) erros aplicados na formulação e funcionamento da interface do sistema;
c) presença de possíveis erros em banco de dados ou em estruturas de dados;
d) baixo desempenho ou erros durante a execução;
e) possíveis erros no inicio e termino do sistema.

O teste de Caixa Preta como dito anteriormente não necessita de conhecimento a estrutura de código para que seja realizado, porem dentro dessa técnica de teste, existe o que

seriam sub-técnicas que fazem parte desse teste, onde juntas formam o teste de Caixa Preta ou funcional (PETERS; PEDRYCZ, 2001).

5.1 AS SUB-TÉCNICAS DE CAIXA PRETA

Uma das sub-técnicas que pertencem ao teste de Caixa Preta é a Analise de Valor Limite, levando em consideração que a maior quantidade de erros ocorre nas extremidades, a Analise de Valor Limite não resolve os erros encontrados no interior das classes, mas sim levam eles para as extremidades das classes para serem resolvidos juntamente com os outros (PRESSMAN, 2006).

O teste orientado por sintaxe é outra sub-técnica existente no teste de Caixa Preta, esse tipo de teste é executado sobre as gramáticas que são desenvolvidas em sistema. A maior parte desse tipo de teste é aplicado a compiladores, pois como foi dito tem a propriedade de trabalhar sobre a gramática descrita no programa, a forma de execução é feita onde, após descrita as especificações, o analisador testa de maneira que cada regra da produção criada seja aplicada pelo menos uma vez (PETERS; PEDRYCZ, 2001).

O teste de comparação que é uma sub-técnica pertencente ao teste de Caixa Preta, certamente em alguns casos ouviu-se em falar em confiabilidade de um sistema, um exemplo onde isso se torna necessário é em sistemas de controle aéreo, na busca de uma confiabilidade maior, muitas vezes são criados sistema redundantes, mas com versões diferentes. O teste de comparação executa esses sistemas redundantes com resultados obtidos em tempo real, assim avaliados as suas repostas e confirmando a sua confiabilidade e consistências (PRESSMAN, 2006).

O teste Baseado na Tabela de Decisão é uma sub-técnica que pertence ao teste de Caixa Preta, esse teste trabalha de forma nos requisitos criados anteriormente em forma de

instruções, essas instruções que o software possui dentro de seu código fonte também é conhecida como regras. As regras são avaliadas de maneira estrutural onde cada uma delas é aplicada pelo menos uma vez durante o teste. A Tabela de Decisão possui em sua composição os números de colunas com todas as possibilidades que o teste oferece, porem no tipo da tabela fica as condições que devem ser realizadas, no final da tabela se encontra os resultados das regras (PETERS; PEDRYCZ, 2001).

A técnica de grafo de causa e efeito também é uma sub-técnica pertencente ao teste de Caixa Preta, muitas vezes ocorrem erros na interpretação do engenheiro no momento de transcrever um algoritmo ou um programa que se encontra em pseudocódigo para a linguagem determinada, o teste de caso efeito busca traduzir esses algoritmos em algo de melhor compreensão, para que essa técnica ou sub-técnica seja realizada é preciso seguir quatro passos (PRESSMAN, 2006):

a) causa são as condições de entrada que é presta no teste, efeito são as ações realizadas no teste. As causas e os efeitos são aplicados em execuções distintas uma da outra;
b) a melhor interpretação do engenheiro solicita então que ocorra a criação de um grafo de causa e efeito, dessa maneira o grafo é desenvolvido;
c) uma tabela de decisão então é criada com o resultado da conversão do grafo de causa e efeito;
d) os casos de teste são adquiridos pela conversão da tabela de decisão.

O uso de uma tabela de decisão no acarreta somente vantagens, mas também possui as suas desvantagens, pois as entradas não são consideradas juntas e sim separadamente, mesmo que se necessite de outra forma de realizar os testes (PETER; PEDRYCZ, 2001).

5.2 PARTICIONAMENTO DE EQUIVALÊNCIA

O particionamento de equivalência trabalha dividindo as entradas de dados em classes que possuem as mesmas características, para que desta maneira o teste possam localizar então erros que possivelmente surgem de um mesmo tipo de dados. Utilizando essa maneira de busca e execução do teste de Caixa Preta, possui o objetivo de reduzir a quantidade de erros isolados, pois transformando eles em um conjunto o teste pode ser realizado de uma maneira consideravelmente mais rápida e eficiente (PRESSMAN, 2006).

A divisão de classe para a realização de testes possui como base a avaliação das classes e com isso identificar uma condição de entrada, para essas classes observa-se também que em uma classe de equivalência há um conjunto de estados que se caracterizam de duas formas, válidos e inválidos para ser uma condição de entrada. Uma condição de entrada muitas vezes podem ser valores numéricos, um conjunto que possui valores com as mesmas características ou mesmo um intervalo de valores. As classes de equivalência possuem algumas diretrizes que se deve respeitar para que elas sejam definidas (PRESSMAN, 2006):

a) a ocorrência de uma condição de entrada seja um intervalo, então é definida uma classe de equivalência valida e duas classes de equivalências invalidas;
b) caso a condição de entrada seja um valor especifico, ocorre o mesmo que na diretriz anterior, uma classe de equivalência valida e duas classes invalidas;
c) se houver a ocorrência de um conjunto, então haverá a definição de uma classe de equivalência valida e uma classe de equivalência invalida;
d) se houver uma condição de entrada booleana, então acontecerá o mesmo citado anteriormente, uma classe de equivalência valida e uma classe de equivalência invalida.

A identificação de uma partição de equivalência ocorre pelas especificações que o programa possui, também pode aparecer à identificação de uma partição equivalente na documentação que o usuário tem em suas mãos. O Engenheiro identifica a partição pela sua experiência de observar essas classes de valores de entrada, pois elas também possuem uma probabilidade de conter erros, um bom exemplo para observar esses erros é utilizando um programa que possui em seu código uma rotina de busca, pois dessa maneira a forma de trabalhar do software possibilita a procura de uma seqüência de elementos, onde dentro dessa seqüência um elemento será retornado como se fosse um elemento chave, de acordo com a Figura 3 (SOMMERVILLE, 2003).

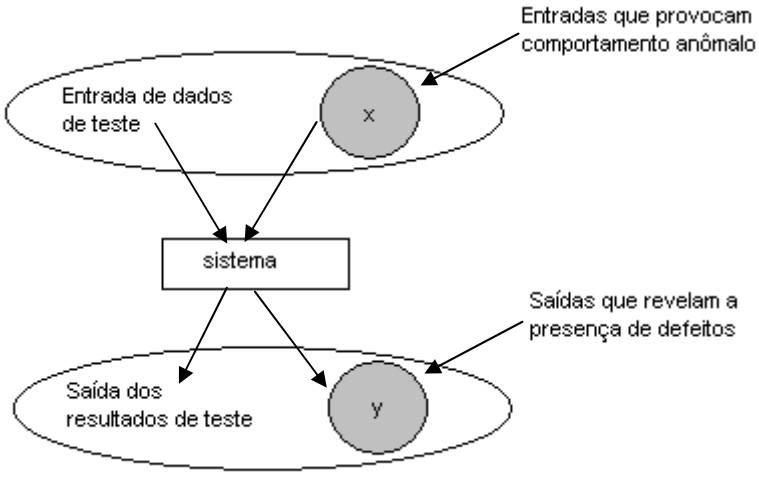


Figura 3. Parcionamento de equivalência
Fonte: SOMMERVILLE, I (2003)

As rotinas que são utilizadas na partição de equivalência possuem especificações. A especificação de precondição trata que na realização do trabalho da rotina não poderá possuir uma seqüência de elemento vazia. A pós-rotina estabelece a declaração de uma nova variável chamada *found*, caso ocorra à verificação e comprovação da posição do elemento chave dentro da seqüência de elementos (SOMMERVILLE, 2003).

As saídas que o sistema mostrara são muito importantes, pois são com elas que o Engenheiro de Software poderá observar se o sistema executara corretamente, caso ele mostre

alguma saída diferente daquela que deveria mostrar é porque possui algum defeito em sua funcionalidade.

A princípio esse teste de funcionalidade é realizado de forma manual, isso pode leva um tempo considerável, pois dessa maneira o engenheiro necessita que cada função do programa seja executada para observar se elas realmente estão operando de forma correta.

O ideal é a construção de uma ferramenta que realizasse o teste funcional para o testador, auxiliando na busca de erros dentro de um sistema, essa ferramenta pouparia tempo, pois realizaria o teste mais rapidamente, também pouparia serviço ao testador que fica livre de fazer o teste manualmente podendo assim realizar outra atividade.

6 FERRAMENTA I&T MANAGER (ISSUE AND TEST MANAGER)

A ferramenta desenvolvida possibilita o cadastramento das características necessárias para o teste de software, com a inserção de informações no sistema, o usuário desenvolve o seu plano para uma orientação de trabalho durante a execução do teste. Com o plano desenvolvido cria-se a condição para executar o mesmo com a integração do programa a um sistema de acompanhamento de defeitos.

A criação de uma ferramenta de teste com essas características pode ajudar os testadores de software na hora de realizar seus testes, essa provável ajuda acontece devido a sua propriedade de trabalhar direto no sistema especificando as entradas de dados e mostrando as saídas, para que posteriormente possam ser avaliadas pelo Engenheiro de Teste.

6.1 METODOLOGIA APLICADA NO SISTEMA

A metodologia utilizada para a elaboração do trabalho possui características que tem a função de destacá-la perante outro método de trabalho ou execução de sistema. O controle de testes nos softwares desenvolvidos visão a garantir a qualidade e conformidade dos dados e informações processadas, evitando problemas futuros na sua utilização pelo cliente. A falta de testes ou uma metodologia falha no planejamento, execução ou até mesmo na análise, pode por em risco o trabalho de desenvolvimento, podendo gerar prejuízos de diversas ordens, como retrabalho e ônus de credibilidade da empresa desenvolvedora de software entre outros. Um sistema falho pode também, gerar problemas de ordem financeira, como informação incorreta, atrasos de entrega, valores incorretos ou mesmo de ordem material e humana.

Uma metodologia consistente e auxiliada por uma ferramenta que garante a otimização e controle dos testes, é fundamental para tornar o processo de testes mais eficiente. A ferramenta proposta neste trabalho visa, alem do controle de teste de sistemas e programas de computador, orientar os gerentes, analistas e usuários a seguir uma metodologia de testes, além do apoio a analise e gerenciamento de resultados. Com propósito de diferencial competitivo, a ferramenta destina-se ao cumprimento das regras básicas da garantia da qualidade, proporcionando a melhoria constante dos processos que envolvem o desenvolvimento de programas e sistemas computacionais.

O ambiente de execução da ferramenta foi criteriosamente analisada e definida com base na crescente utilização da internet. Desenvolvida na linguagem *Java,* torna-se possível a utilização em qualquer servidor internet que rode programas *Java* ou pela rede local onde os computadores estão ligados (intranet). A possibilidade da instalação em um servidor onde vários usuários terão a possibilidade de acessar o mesmo programa, tornando assim uma aplicação que pode possuir seus resultados integrados, facilitando a verificação de relatórios de trabalhos e execução de teste pelo gerente de teste.

O uso de uma ferramenta que possui essa característica de funcionamento pela rede, possibilita uma mobilidade e uma versatilidade maior perante uma ferramenta de características *desktop*, onde esta fica instalada em um único computador.

O sistema que possui essa característica trabalha individualmente, executando e armazenando os dados dentro de sua memória. A instalação é única para cada computador, assim rodando suas funções e especificações em um determinado lugar, dificultando uma provável integração ente o resultado de duas ou mais máquinas. A individualidade de execução de testes desses computadores cria relatórios específicos para cada resultado, diminuindo o compartilhamento das informações prejudicando a mobilidade dos dados,

resultando na dificuldade criação de um único relatório completo com todos os dados e todos os usuários.

A mobilidade dessa metodologia possibilita então que vários usuários tenham acesso ao sistema ao mesmo tempo, sendo que cada um deles vai possuir um perfil. A criação das regras de perfis definirá as limitações das ações que serão executadas no sistema, para que o isso seja possível, são utilizados papéis para caracterizar cada grupo de usuários que manipulam as informações que são inclusas deletadas e consultadas no programa.

A utilização das regras influencia todas as atividades que o sistema executará, porque é no uso dessa filtragem que a barra de *menu* vai disponibilizar as funções para o usuário. As operações do sistema são todas exercidas nesta barra, sejam elas para cadastro ou consulta. Nesta barra também esta contida as atividades que o programa executará, onde essas são limitadas pelo perfil de usuário, igualmente como feito com o cadastro.

Uma atividade da ferramenta é a integração do teste em execução com uma ferramenta de depuração (*debug)*, onde desta será retirado código do erro ou falha (*bug)* ocorrido durante o teste. O *bug* é um erro que pode ocorrer durante a utilização ou desenvolvimento de um software, sendo que esse erro pode ter conseqüências diferentes, como mostrar algo que aconteceu e não deveria acontecer com esse programa ou ate mesmo fechar o programa, nessa condição podendo perder as informações que estão sendo utilizadas no momento.

6.1.1 Perfis dos Usuários

A definição dos papéis é importante para a caracterização das limitações que cada usuário vai possuir dentro do sistema, essa especificação é feita pela função exercida na corporação onde trabalha. Existem algumas regras de acesso que podem ser referidos as

pessoas que farão parte do sistema. Inicialmente o sistema conta com três papéis básicos, definidos no sistema como perfil de usuário: Gerente de testes, Analista de testes e Testador. Estas funções são definidas no sistema pelo perfil do usuário, desta forma se o utilizador do sistema necessitar de mais papéis específicos, estes serão criados a partir do cadastro de perfil e das liberações das funcionalidades do sistema a este perfil. O cadastramento incorreto poderá liberar funcionalidades ao usuário, dos quais ele não deve ter acesso, portando, esta parte do sistema deve ser utilizada com responsabilidade, preferencialmente por cargos gerenciais. A criação de perfis pode ser aumentada se a organização preferir, criando-se mais papeis no sistema, liberando-se os devidos acessos aos perfis.

O primeiro papel que pode ser definido é o Gerente de Teste, onde esse usuário terá acesso pleno em todas as atividades, podendo criar planos de teste, cadastrar teste, cadastrar usuário, permitir liberação de perfil para o usuário, relatórios e execução dos testes, o Gerente de Teste terá todas as condições para administrar o sistema.

O segundo papel que pode ser definido é o analista de teste. O usuário que possui essa característica carregará algumas limitações, uma delas é a criação e cadastramento de um novo plano de teste, essa função fica exclusivamente para o Gerente. O Analista de Teste pode realizar alguns cadastros e monitoramento dos testes realizado, assim verificando o andamento do teste e a produção do Testador.

O terceiro papel a ser definido é o Testador, a limitação desse usuário dentro do sistema é total, ele não terá acesso a cadastro e nem consultas, unicamente ele entrará no sistema e receberá as informações necessárias para a execução do teste no qual ele foi designado, as suas atividades e os passos que ele deve seguir estarão no plano de teste, onde essas informações foram cadastradas anteriormente pelo Gerente de Teste. O usuário não terá permissão para criar planos de teste, cadastrar usuário, cadastrar perfil e gerenciamento de teste, como procedimentos e atividades.

A Tabela 1 mostra a tabela de usuário e de perfis.

	Gerente de Teste	Analista de Teste	Testador
Administração	Aplica	Auxilia	-
Gerencia de Teste	-	Aplica	-
Execução de Teste	Aplica	Aplica	-
Resultados	Aplica	Aplica	-

Tabela 1. Tabela de usuários e de perfis

A utilização desses papéis pela ferramenta é feita de maneira integra, não permitindo assim que um usuário de perfil mais limitado consiga entrar no sistema de alguma forma e alterar sua condição dentro da regra do sistema para uma menos limitada ou mesmo ilimitada podendo assim realizar funções que não são destinadas a sua realidade.

6.1.2 Ferramenta de Apoio à Metodologia

A ferramenta possui acesso às funcionalidades através de uma barra de opções (*menu*) que possibilita a navegação entre as funções do sistema, permitindo a particularidade de trabalho conforme as características de cada perfil, assim disponibilizando apenas os acessos necessários ou pertinentes para cada tipo de usuário.

O *menu* disponibiliza ao usuário todas as funções para que ele realize suas atividades com a ferramenta, sendo desde o cadastro e se estendo as consultas ou relatórios. A disponibilidade para a utilização do mesmo é avaliada no momento em que o usuário que executará o sistema, entra com o seu usuário e sua senha, dessa forma o programa verifica as suas características liberando ou não as funções.

As funções apresentadas pelo sistema são quatro:

a) **administração**: essa opção possibilita ao usuário cadastrar ou editar perfis e usuários, sendo que respeita a limitação do *login* que está executando o sistema naquele momento;

b) **gerencia de Teste**: esse *menu* possui dentro dele mais doze *submenus*, que permite o cadastro e edição de suas atividades, as oito opções desse menu são: Procedimento de Teste, Caso de Teste, Atividade de Teste, Ciclo de Vida, Ferramenta de Teste, Ambiente de Teste, Domínio de Teste, Modulo;

c) **execução de Teste**: aqui o usuário poderá ter acesso às opções necessárias para a execução dos testes, as alternativas que estão dentro desta função são: plano de teste e execução de teste, mas para que seja liberado o acesso nestas opções é necessário que o usuário tenha um perfil autorizado para esse nível de trabalho;

d) **relatório**: na opção de relatório, tense a possibilidade de visualizar os resultados e métricas de testes, obtidos com a execução dos testes de *software*, mas somente poderá acessar essa função quem possuir um perfil com acesso. Os relatórios permitem a observação de todas as etapas e característica que foram utilizadas para a realização do teste em um determinado programa. As métricas auxiliam a elaboração do nível de qualidade e orientam o desenvolvimento e correção de funcionalidades.

A administração é a opção que possui a função de cadastro de usuário e de perfil, na opção de cadastro será descrito o nome do usuário, o *login*, o e-mail, perfil que receberá, as limitações dentro do sistema de teste e o seu status que poderá ser ativo ou inativo. O perfil é cadastrado para indica limitações que ocorre dentro do programa, ele ainda está ligado ao submenu de usuário, porque é com a ajuda dessa opção que cada uma das pessoas que são cadastradas tem suas regras de acesso, como visto na figura 4.

Figura 4. Imagem exemplificando a opção Administração

O gerenciamento de teste é constituído pelas opções necessárias para a criação de um plano de teste e também para sua execução, a primeira informação que se solicita ao usuário é o cadastro do procedimento de teste, isso nada mais é que o tipo de teste que será executado para determinado sistema. O caso de teste é caracterizado pela descrita de como ocorrerá o teste dentro do sistema, significa passo a passo o deve ser feito pelo testador durante a execução do teste. A atividade de teste indica qual tarefa deve ser exercido no determinado momento, mostrando para o usuário o que falta para ser realizado. O ciclo de vida do teste é necessário para armazenar em que situação ele se encontra. A Ferramenta de teste cadastrará a ferramenta *debug* que o usuário poderá escolher na realização dos testes, isso é possível porque o I&T Manager (*Issue and Test Manager*) é um sistema de integração, para interagir o teste a uma ferramenta de erros ou uma *Issue Track*. O Ambiente possibilita ao testador o cadastro de um ambiente de sistema para ele realizar suas funções, esse ambiente indica qual sistema operacional que o computador ou o programa a ser investigado está rodando. O Domínio possibilita escolher qual região do programa passará pelo teste, assim permitindo a divisão em setores para sua execução. O modulo permite cadastrar a técnica que o usuário usará para realizar o teste. A Figura 5 apresenta um exemplo desta opção do menu.

Figura 5. *Menu* Gerência de Teste

Execução de Teste é a opção onde ocorre o planejamento e a execução dos testes. Com a criação do plano de teste, um roteiro de teste é concebido, conforme as características escolhidas pelo gerente ou analista de testes. Nesta etapa defini-se o cronograma, casos de teste a serem testados ou não, programa, ferramentas a serem utilizadas, enfim, toda a massa de testes. Também são distribuídas as atividades aos testadores, que na opção de "execução" realizam as tarefas planejadas, registrando no sistema os resultados dos testes e eventuais falhas e erros (*bugs*).

Esses registros são fundamentais para correção do programa, alem de medir a qualidade, é fundamental para a equipe de desenvolvimento, que utiliza os resultados para melhoria do sistema, principalmente nas falhas. Resalta-se que o planejamento dos testes é fundamental para a qualidade dos resultados, como mostra a Figura 6.

Figura 6. Opção Execução de Teste da ferramenta

Na opção de relatórios têm-se as métricas dos testes. O gerente e o analista de testes podem acompanhar os resultados dos testes, analisarem gráficos com tempos ou mesmo o numero de erros encontrados. Alem da qualidade do programa em fase de testes, pode-se através dos relatórios, verificarem os históricos de testes e erros encontrados, exemplificado abaixo pela Figura 7.

Figura 7. Possibilita a escolha da impressão de relatórios

O sistema permite gerar alguns gráficos para os relatórios dos dados imputados nos testes. Inicialmente é possível gerar os relatórios de qualidade do sistema em teste, como: defeitos encontrados, horas de teste e resultados das execuções dos testes. Com a análise das falhas e defeitos, pode planejar as correções e novos testes, como mostra a Figura 8.

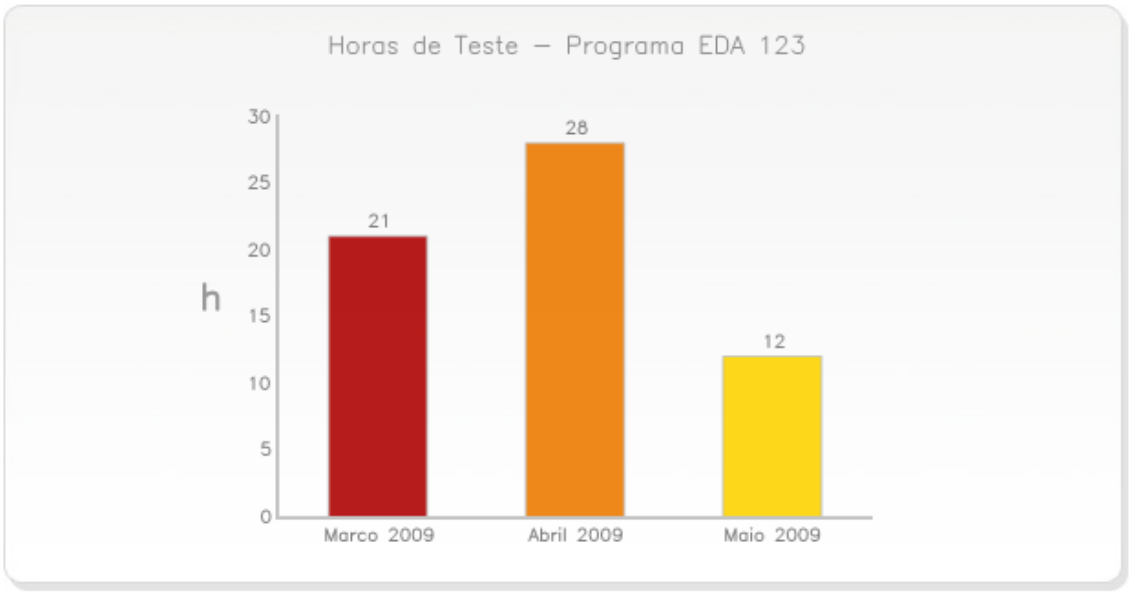


Figura 8: Numero de horas de teste por programa

Os defeitos podem ser futuramente classificados por prioridade e com a integração com outra ferramenta de acompanhamento, torna-se o processo mais de controle de falhas mais eficiente. O defeito é cadastrado na ferramenta de apoio e esta transporta a falha para ferramenta específica de bugtrack.

O gerente de testes também pode avaliar sua equipe de teste, através das analises dos testes planejados e executados. Programas detentores de um número maior de defeitos e falhas poderão receber maiores numero de horas para novos testes, conforme a Figura 9.

Figura 9: Horas planejas e Realizadas em teste

A equipe de desenvolvimento poderá usar os mesmos relatórios para orientar seus funcionários a agilizar a correção de falhas mais graves ou os programas com maior número de falhas, assim demonstra a Figura 10.

Figura 10. Defeitos detectados por programa testado

Alem dos gráficos, todos os resultados podem ser listados, com os comentários dos testadores.

A opção Ajuda oferece ao usuário um suporte para que ele tenha condição de operar as funções que o sistema disponibiliza, sendo que o auxilio é obtido com a utilização de *links* explicativos sobre cada função da ferramenta, de acordo com a Figura 11.

Figura 11. Opção de Ajuda da ferramenta

O *menu* é o principal meio de comunicação entre o usuário e o sistema, isso permite que o sistema tenha um controle nas funções, nas quais o usuário poderá fazer uso no momento de executar um teste em um determinado sistema.

6.1.3 Ferramenta de Depuração de erros

As ferramentas de *debug* ou *issue tracks* são ferramentas que possuem características de localização de erros em um sistema, elas possibilitam então encontrar um erro em um determinado sistema, para melhorar a varredura de uma ferramenta em um sistema é criada uma metodologia que posteriormente se torna uma ferramenta de integração.

A I&T Manager é uma metodologia que integra o sistema que está sendo testado a uma *Issue Track*, essa integração possibilita ao testador a verificação, obtendo dessa forma o código do erro que aconteceu durante o seu teste, a ferramenta que receberá essa função de capturar os erros é o *Mantis*, mas essa ferramenta não é a única que está no mercado com essas características, além dela existem outros programas que executam essas funções, entre os outros programas estão o Bugzilla e o Trac.

O *Mantis* é uma Ferramenta gratuita classificada para o uso via *web*. Ela foi desenvolvida em *Php* e interage com o banco de dados *Mysql*, possui distribuição para *Windows, Mac OS*. O seu símbolo é uma homenagem a um inseto chamado Louva-Deus, seu criador se chama Victor Boctor, esse sistema ainda possui distribuição em vários idiomas inclusive o português brasileiro e sua licença é *GNU* (*General Public License*), possuindo uma longa lista de recursos, entre esse recursos estão alguns bancos de dados nos quais ele pode registrar suas informações. A Figura 12 mostra um exemplo da ferramenta de erros Mantis.

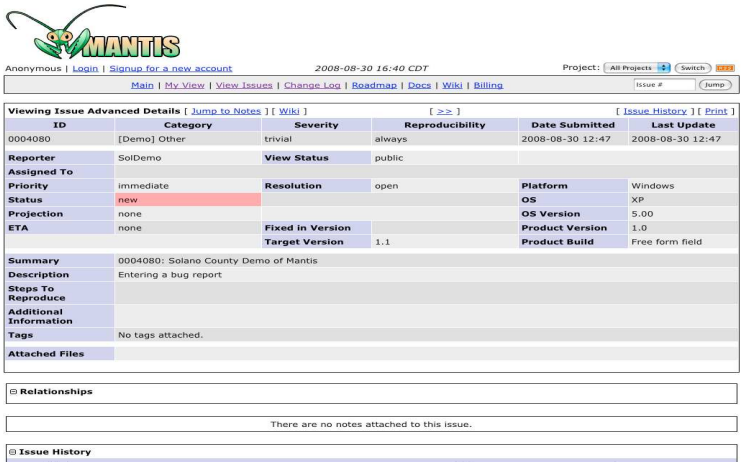


Figura 12. Exemplo do *Mantis*

O *Bugzilla* é um sistema parecido com o *Mantis*, porque igualmente ao anterior ele produz uma varredura no sistema que está sendo testado para procurar erros que podem estar contidos dentro de sua funcionalidade. O sistema de teste foi desenvolvido em uma linguagem de programação chamada *Pearl* e suas funções são realizadas via web, o sistema *Bugzilla* é um *software* livre, dessa maneira vários usuários podem acessar seu código fonte e alterar para encontrar uma melhor situação de trabalho para suas atividades. A manutenção desse programa não é realizada por uma única pessoa, mas sim com o auxilio de vários usuários que sempre buscam melhorar a ferramenta.

O funcionamento do programa não é complicado, levando em consideração que o usuário preencherá os campos solicitados pelo sistema, aglomerando essas informações a ferramenta realiza o teste de verificação de erros, como na Figura 13.

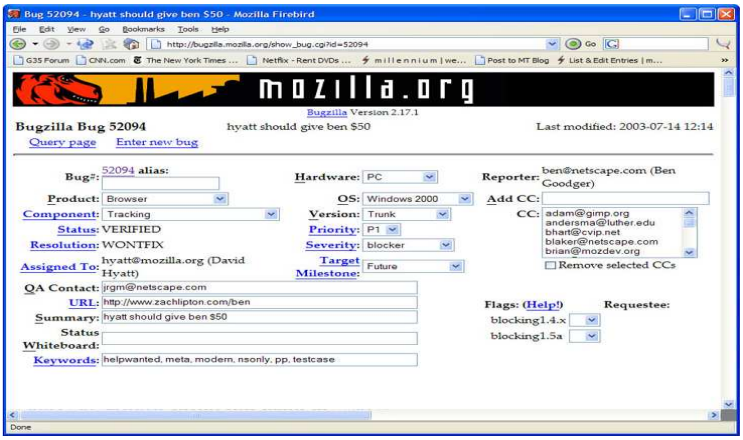


Figura 13. Imagem do *Bugzilla*

O trac é uma ferramenta de busca de erros, ela possui a função de observar possíveis discordâncias entre aquilo que o programa está executando e o que ele realmente deveria executar ou proceder. O sistema segue o padrão das outras duas ferramentas e igualmente está no grupo de sistema livre onde seu código é aberto para que outras pessoas possam observar seu funcionamento. A diferença que persiste entre as três ferramentas é a linguagem nas quais elas foram desenvolvidas, o *Trac* por sua vez foi criado em uma linguagem que chama *Python*, uma linguagem nova no mercado e que vem a acrescentar em muito na sociedade dos desenvolvedores. A Figura 14 apresenta um exemplo da interface da ferramenta.

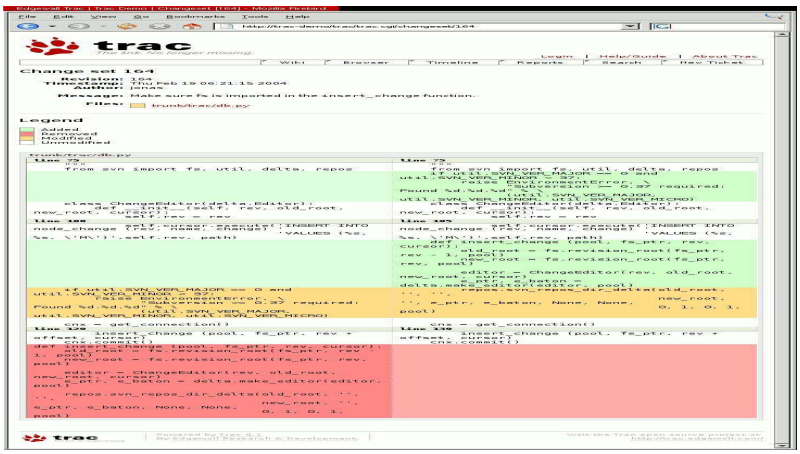


Figura 14. Característica da ferramenta *Trac*

As ferramentas de *debug* como são conhecidas esses sistemas, são muito interessante para o desenvolvimento de testes automatizados, porque elas permitem a criação de condições onde se leva o programa no qual se deseja testar a uma situação que possível ocorrência de erros.

6.1.4 A Linguagem de Modelagem Unificada (UML)

A linguagem de Modelagem Unificada (UML) é uma forma gráfica de apresenta projetos de sistemas que estão em desenvolvimento. A UML possui vários formatos para

representar os projetos que estão em fase de desenvolvimento, um deles á a utilização do símbolo retangular quando deseja representar uma classe de dados projeto (BRAUDE, 2005).

A UML trabalha com a realização de uma forma gráfica de sistemas de complexidade elevada e que possuem a necessidade de serem representados graficamente antes de serem implementados pelos desenvolvedores (BOOCH; RUMBAUGH; JACOBSON, 1999).

A utilização dessa linguagem na modelagem da ferramenta é importante, pois assim é possível observar como operará a ferramenta com todos os seus atributos e funções, isso antes de realizar a implementação.

A modelagem do sistema é necessária para que ocorra um estudo anterior à implementação da ferramenta, esse estudo efetuado da ferramenta mostrará as funções que ela receberá durante a sua implementação. A Figura 15 mostrará um exemplo da Modelagem da Ferramenta.

O ArgoUML é um sistema que possibilita a criação dos modelos UML. A partir deste programa o desenvolvedor cria seus modelos.

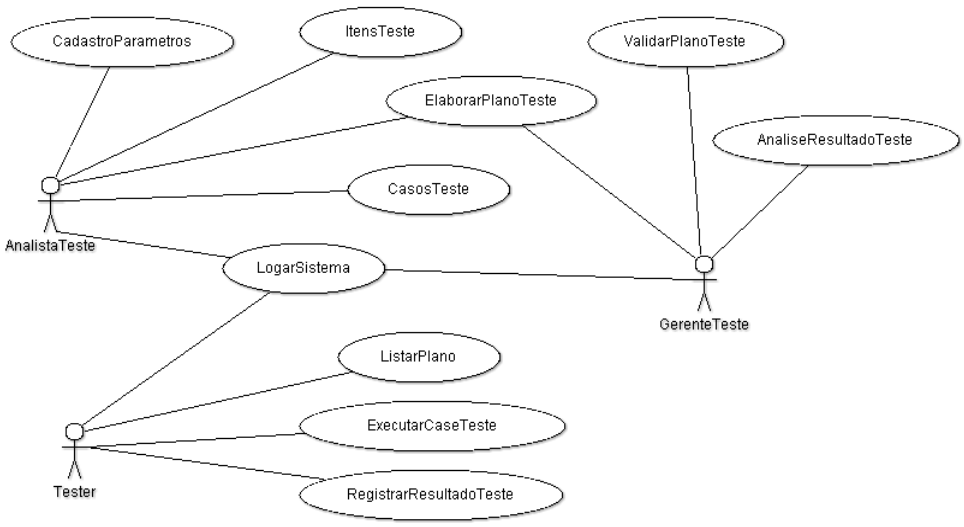


Figura 15. Modelagem Use Case da Ferramenta

A modelagem da ferramenta utilizando o Use Case se caracterizou devido a sua propriedade de mostrar de maneira mais clara a função de cada papel dentro do sistema desenvolvido.

6.1.5 A Linguagem de Desenvolvimento JAVA

A linguagem Java é utilizada para reverter ações humanas em ações computacionais, ela é uma linguagem que possui várias plataformas para sua aplicação, desde unidades móveis até servidores (DEITEL; DEITEL, 2003).

A escolha da linguagem Java para a criação da ferramenta ocorre devido a sua variedade de plataforma de trabalho, sendo a IDE escolhida para o desenvolvimento o Netbeans, sendo uma ferramenta de criação de software que utiliza a linguagem de programação JAVA.

6.1.6 Modelagem da Base de Dados da ferramenta

A Base de Dados é parte fundamental em um sistema computacional, porque é nela que ficam guardadas todas as informações que a ferramenta obterá com a sua execução. Existem muitos programas com características para o desenvolvimento de um Banco de Dados, mas o utilizado para a criação da base desse sistema foi o *SQLyog*.

O *SQLyog* é uma ferramenta de desenvolvimento e manipulação de Banco de Dados *Mysql*, sendo que a linguagem utilizada para esse programa é o *SQL*. O programa possibilita ao usuário a criação da modelagem da base de dados. A Figura 16 mostra um exemplo da modelagem elaborada para o banco deste sistema.

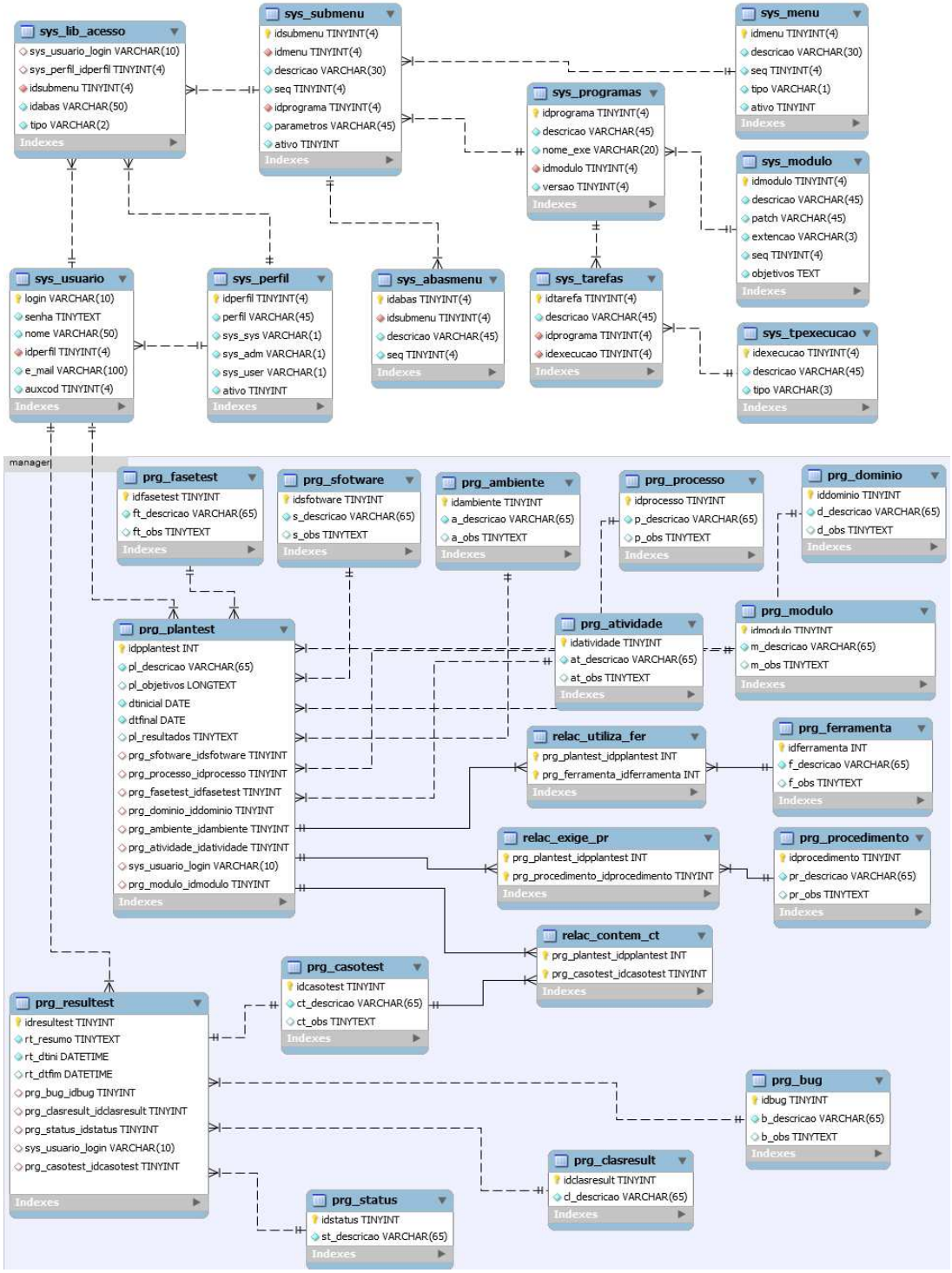


Figura 16. Exemplo da modelagem do Banco de Dados do sistema

A modelagem observada foi inteiramente elaborada na ferramenta *SQLyog* como o banco de dados, dessa forma ocorre a junção entre as duas situações.

6.1.7 Resultados Obtidos

A realização da pesquisa contribuiu para o ganho de conhecimento na área de Engenharia de Software, com esse aprendizado, pode-se então entender de forma clara e objetiva o funcionamento da metodologia de realização de testes de software.

O estudo voltado para os testes de software propiciou a elaboração de um plano de desenvolvimento para uma ferramenta, que realiza essas avaliações de forma automática ou semi-automática. O sistema de integração possibilitou a união ente dois softwares que estão sendo utilizados para o mesmo fim, ou seja, um para ser testado e outro para ser avaliador.

A ferramenta conseguiu ser segura para a utilização, devido a sua função de verificação dos limites que cada usuário possui, analisando assim a função desempenhada pelo usuário que entrou no sistema naquele determinado momento.

A execução do testes permitiu que fossem cadastradas diversas características necessárias para a criação de um plano de execução de teste, dessa forma pode-se avaliar e acompanhar a trajetória de um determinado erro.

Chegou-se a conclusão que, a ferramenta pode atender aos parâmetros nela especificados, assim possibilitando a utilização de um sistema que atenda a praticamente todas as necessidades criada em um ambiente de teste, assim auxiliando na melhoria do desempenho da equipe que usá-lo como opção de trabalho.

6.2 METODOLOGIA APLICADA NO DESENVOLVIMENTO DO TRABALHO

A metodologia ajuda em uma melhor organização do trabalho, dividindo a pesquisa em etapas e colocando-as em uma ordem a ser seguida, dessa forma é possível que

ocorra uma melhor execução das etapas de pesquisa sem que ultrapasse o tempo definido no cronograma.

6.2.1 Estudo de Testes

O estudo de teste se fez necessário para obter um melhor entendimento de como funciona o teste de software.

A pesquisa sobre o conhecimento da área efetuou-se em busca realizada em livros correspondente ao assunto em questão, onde se podem encontrar informações que levassem a construção do projeto de pesquisa.

O estudo dos testes trouxe a compreensão necessária para o desenvolvimento do capítulo explicativo do assunto, pois na busca pelo conhecimento nesta área foi possível a compreensão de várias características pertencente aos testes.

6.2.2 Estudo voltado a Técnica de Caixa Preta

O estudo da técnica de Caixa Preta foi viabilizado pela pesquisa realizada em livros que estudam as técnicas de testes de software.

A realização de busca nesses livros ocorre, pois a técnica de Caixa Preta se trata de umas das técnicas de testes realizado em software, dessa maneira primeiramente se faz necessário o conhecimento voltado aos testes para posteriormente conhecer as suas técnicas e ramificações pertencentes a essa área de avaliações.

A Caixa Preta é uma técnica de teste onde sua realização ocorre de forma funcional dentro de um sistema.

6.2.3 Levantamento dos requisitos para a Ferramenta

O levantamento dos requisitos para a ferramenta ocorre em livros que tratam de testes de software e das técnicas, pois nelas que são feitas às restrições que o sistema deve possuir para que o teste seja realizado de maneira eficiente.

Os requisitos adquiridos nos estudos são aplicados na construção da ferramenta, onde fazendo uso desses requisitos possa avaliar o sistema, identificando a ocorrência de erros no software avaliado.

6.2.4 Modelagem da Ferramenta

A modelagem da ferramenta acontece com o uso de uma ferramenta de UML, para a modelagem ocorre à necessidade de pesquisa sobre a área em livros que tratem desse tipo de linguagem com suas características e funções.

6.2.5 Implementação da Ferramenta

A implementação do sistema acontece com a utilização da linguagem de programação Java. A utilização da linguagem Java para a implementação do software trouxe a necessidade de realizar pesquisas sobre a linguagem em livros referentes.

A busca de livros sobre o assunto é importante para sanar dúvidas que muito provavelmente acontecem durante a construção de um sistema.

6.3 RECURSOS NECESSÁRIOS

Os recursos são aquilo que se fazem necessário para a realização do projeto de pesquisa e conseqüentemente da ferramenta proposta anteriormente. Os recursos para a construção da ferramenta é:

a) *hardware*: os recursos de *hardware* são os recursos físicos que são necessários para a construção da ferramenta, no caso é necessário um microcomputador com processador *core 2 duo* e memória de 1024 *Gigabytes*;

b) *software*: esse tipo de recurso envolve as necessidades de suprimento de programas para que se consiga desenvolver o projeto de pesquisa. O recurso de *software* envolve desde a linguagem que será utilizada para a modelagem da ferramenta até a linguagem na qual ela será desenvolvida. No caso se possui a necessidade de uma ferramenta de UML para a modelagem e uma versão gratuita da linguagem *Java* para o desenvolvimento da ferramenta. O sistema necessita na sua execução o *Apache* para rodar em ambiente *web*.

A definição dos recursos necessários é importante o planejamento da implementação e execução de um sistema.

7 TRABALHOS CORRELATOS

Os trabalhos correlatos são trabalhos que já foram desenvolvidos e que possui alguma ligação com o projeto de pesquisa que está sendo desenvolvido.

Os trabalhos podem ter ligação desde a área de testes até mesmo trabalhos que falam sobre ferramentas de teste criadas utilizando a técnica de Caixa Preta.

7.1 TESTES

A área de teste pode possuir maneiras de expressar a forma de realização de testes, mas todas levam para uma mesma afirmação, onde diz que testes de software é a realização do sistema de maneira que se possa controlá-lo, observando se a sua execução atinge o objetivo especificado. A transformações que um software pode envolvido faz com que a realização de testes de software se torne algo não comum, algumas características que agregam a isso são (JINO et al, 2004):

a) processo com alto valor de custo;
b) falta de conhecimento envolvendo o custo beneficio de realizar testes em software;
c) falta de mão de obra especializada na área de realização de testes;
d) não conhecimento de um teste ideal para a situação em questão.

A realização de testes consiste em buscar erros em software, tendo o objetivo de melhorá-lo, onde esses testes são realizado nas etapas de desenvolvimento dos sistemas para que se encontre prováveis erros o mais cedo possível, para que sejam resolvidos e o software siga o seu desenvolvimento operando de maneira que sua execução está correta (CUNHA et al, 2004).

7.2 TÉCNICA DE CAIXA PRETA

O teste de Caixa Preta possui a função de realizar testes funcionais com os sistemas que estão sendo avaliados, testando a sua funcionalidade e não a sua implementação (BENITTI; ZIMMERMANN, 2007).

O teste de Caixa Preta divide as entradas e saídas definindo assim uma abrangência satisfatória quando define uma cobertura do sistema, sendo que dentro do teste de Caixa Preta existem quatro etapas que são utilizadas para a realização dos testes que são: partionamento de equivalência, teste de comparação, análise de valor limite e técnica de grafo de causa e efeito (BENITTI; ZIMMERMANN, 2007).

A utilização dessas etapas isoladamente não proporciona o resultado esperado, dessa maneira o ideal é a utilização dessas atividades juntas para uma eficaz avaliação no software a ser testado.

8 CONCLUSÃO

Os estudos voltados na área de Engenharia de Software trouxeram informações que acrescentaram na compreensão das áreas onde ela atua, esses conhecimentos possibilitaram o desenvolvimento de um trabalho pertinente à utilização das funções desempenhadas por essa Engenharia.

A Engenharia de Software é divida em várias etapas, sendo que a fase abordada de maneira mais intensa pelo projeto foi à de teste de software. Os testes são realizados com o auxílio de padronizações caracterizadas pelas técnicas de testes.

As padronizações das técnicas indicam o tipo de teste aplicado em um software, onde a verificação de funcionalidade de um sistema é realizada pelo teste funcional pertencente à Técnica de Caixa Preta. O conhecimento desta técnica auxiliou no desenvolvimento da metodologia da ferramenta de teste devido a sua característica, onde busca conhecer a funcionalidade do sistema sem a necessidade de conhecer seu código fonte.

A utilização de uma ferramenta para o gerenciamento de erros em sistema é necessária para informar ao testador o andamento do processo de um teste, acompanhando assim todos os passos realizados pela ferramenta e verificando as características dos erros que ocorrem na medida em que o processo vai avançando.

A ferramenta integra ainda outro sistema, com isso ela trabalha de maneira automatizada, devido à característica de execução automática do outro sistema, dessa forma ocorre à possibilidade do Engenheiro de Teste observar o resultado final do plano de teste.

O desenvolvimento do projeto realizou-se de maneira integra, atingindo os pontos pré-definidos no início da pesquisa, além de desenvolver de maneira completa o sistema com uma metodologia aplicada a realização de casos de testes.

A metodologia foi utilizada no desenvolvimento da ferramenta de teste I&T Manager. Este sistema permite ao usuário o controle das informações inseridas, contribuindo para uma organização dos dados com um auxilio da execução do plano de teste. A organização desta ferramenta origina-se no início de sua execução, pedindo ao usuário executor do sistema o seu *login* e sua senha, tornando-se uma ferramenta segura para o uso em organizações de desenvolvimento de software. O cadastramento dos usuários e seus papéis dentro da aplicação possibilitam o controle de acesso, esse controle é feito observando a descrição do usuário que busca o acesso.

A Issue and Test Manager é um sistema seguro devido o seu controle de acesso, versátil pelos seus cadastros, sejam eles de usuários ou testes, e organizado pela possibilidade de realizar essas funções no mesmo sistema.

O teste de software é uma área ampla de estudo, onde possibilita-se o desenvolvimento de várias metodologias contribuindo fortemente para a melhoria dos sistemas existentes. O acadêmico disposto a pesquisar essa área possuirá um ganho em seu conhecimento, devido à abrangência e a relevância técnica desse assunto.

9 REFERÊNCIA BIBLIOGRÁFICA

BRAUDE, Eric. Projeto de Software: **Da programação à arquitetura, uma abordagem baseada em Java:** São Paulo: Bookman, 2005.

CRESPO et al. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo**. Centro de Pesquisa Renato Archer (cenPRA). Campinas, 2002.

CUNHA, Cleida A.Q. **Auto *Test* – Um *Framework* Reutilizável para a Automação de Teste Funcional de Software.** III Simpósio Brasileiro de Qualidade de Software. Brasília, 2004.

DEITEL, M.H; DEITEL, J.P. **Java: como programar**. 2 .ed. São Paulo: Bookman, 2003.

INTHURN, Cândida. **Qualidade e Teste de Software.** Florianópolis: Visual Books, 2001.

GUSTAFSON, David A. Engenharia de Software. Porto Alegre: Bookman, 2002.

JINO, Mario et al. **Uma Melhoria para Teste de Software no Contexto da Melhoria do Processo**. III Simpósio Brasileiro de Qualidade de Software. Brasília, 2004.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. São Paulo: Novatec, 2006.

PEDRYCZ, Wiltod; PETERS, James F. **Engenharia de Software – Teoria e Prática**. Rio de Janeiro: Campus, 2001.

PFLEEGER, Shari Lawrence. **Engenharia de Software: teoria e Prática.** 2.ed. São Paulo: Pearson Education do Brasil, 2004.

PRESSMAN, Roger S. **Engenharia de Software.** São Paulo: Pearson Makron Books, 2006.

RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Grady. **UML – GUIA DO USUÁRIO: O mais avançado tutorial sobre Unified Modeling Languagem (UML), elaborado pelos próprios criadores da linguagem.** 3 .ed. Rio de Janeiro: Campus, 1999.

SOMMERVILLE, Ian. **Engenharia de Software.** 6.ed. São Paulo: Addison Wesly, 2003.

ZIMMERMMAN, Ana Paula; BENITII Fabiane Barreto Vavassori. Testware: ferramenta de planejamento e execução de casos de teste. XVI SEMINCO – Seminário de Computação. FURB – Blumenau, 2007.