UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

VINÍCIUS VIEIRA PESSONI

Implantação de Processos de Teste de Software

Um Estudo de Caso voltado ao CERCOMP-UFG

Goiânia
2011

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE TRABALHO DE CONCLUSÃO DE CURSO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Implantação de Processos de Teste de Software – Um Estudo de Caso voltado ao CERCOMP-UFG

Autor(a): Vinícius Vieira Pessoni

Goiânia, 21 de Novembro de 2011.

Vinícius Vieira Pessoni – Autor

Auri Marcelo Rizzo Vincenzi – Orientador

VINÍCIUS VIEIRA PESSONI

Implantação de Processos de Teste de Software

Um Estudo de Caso voltado ao CERCOMP-UFG

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciências da Computação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Área de concentração: Engenharia de Software, Teste de Software.
Orientador: Prof. Auri Marcelo Rizzo Vincenzi

Goiânia
2011

VINÍCIUS VIEIRA PESSONI

Implantação de Processos de Teste de Software

Um Estudo de Caso voltado ao CERCOMP-UFG

Trabalho de Conclusão apresentado à Coordenação do Curso de Ciências da Computação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Bacharel em Ciências da Computação, aprovada em 21 de Novembro de 2011, pela Banca Examinadora constituída pelos professores:

Prof. Auri Marcelo Rizzo Vincenzi
Instituto de Informática – UFG
Presidente da Banca

Prof. Hugo Do Nascimento
Instituto de Informática – UFG

Vinícius Vieira Pessoni

Em sua graduação, pela Universidade Federal de Goiás - UFG participou como organizador e monitor de cursos de extensão em informática com linux para comunidade em geral no ano de 2008. Ainda em 2008, autou como organizador da Escola Regional de Informática do Centro Oeste - ERI-CO. Em 2009, atuou como organizador e monitor de treinamento em suíte BrOffice para o centro de seleção da UFG. Em outubro de 2009, iniciou estágio na área de desenvolvimento de sistemas no Centro de Recursos Computacionais da Universidade Federal de Goiás, função na qual permaneceu até março de 2011. Adicionado a isso, foi integrante do Centro Acadêmico - CA de Computação por dois anos, desempenhando o papel de relações públicas desse. Todas as realizações supracitadas contribuíram grandemente para o seu crescimento pessoal e profissional.

À minha família, base de quem eu sou, e diretivas de quem serei.

Agradecimentos

Agradeço em primeiro lugar a Deus, que possibilitou minha existência privilegiando-me com o dom da vida. Por me dar forças, saúde, inteligência, determinação, e vários outros atributos provindos dele. Por sua misericórdia, fidelidade, pelas promessas cumpridas e pelas que hão de vir.

A minha família - em especial Pai, Mãe e Irmão - por prover um ambiente de amor, carinho, e compreesão. Por estarem presentes em todos os momentos da minha vida, apesar da distância física, sendo meus fundamentos.

Agradeço também ao meu orientador Auri por ser guia neste trabalho e ter se colocado a disposição para atender as minhas dúvidas, contribuido de maneira grandiosa e inenarrável para minha formação. Ao avaliador Hugo que se fez presente por toda minha trajetória acadêmica desde o primeiro ano, me instruindo de várias formas, mesmo não tendo ministrado disciplinas a mim. Ao professor Federson que me ensinou muito mais do que qualquer ementa pode abranger.

Agradeço aos meus colegas de curso, minha melhor amiga, e meus F.R.I.E.N.D.S os quais posso chamar de irmãos, pois se tornaram minha família durante os quatro anos de graduação. Ao lado desses passei momentos inesquecíveis que estarão para sempre em meu coração.

A minha namorada Geovanna, que esteve ao meu lado proporcionando amor, apoio, e principalmente a compreeensão, necessários aos momentos difíceis, e cuja a compania me acrescentou inúmeros momentos felizes.

A todos os professores do Instituto de Informática dos quais fui aluno, aos técnicos administrativos, e a todos que contribuíram de maneira direta ou indiretamente para minha formação e crescimento pessoal, meus sinceros agradecimentos.

"Some of the best lessons we ever learn are learned from past mistakes. The error of the past is the wisdom and success of the future."

Dr. Dale E. Turner,
Escritor e Reverendo Americano.

Resumo

Vieira Pessoni, Vinícius. **Implantação de Processos de Teste de Software**. Goiânia, 2011. 61p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

O assunto tratado por este projeto é a contribuição que a atividade de teste pode trazer no quesito qualidade dos produtos e processos de software. Nesse sentido, o projeto é um estudo de caso voltado a instituição CERCOMP-UFG o qual conta com a construção de uma base de conhecimento para teste bem como a proposta de processos e roteiros adequados à essa atividade específicos para tal instituição.

Palavras–chave

Teste Software, CERCOMP, Processo Teste, Teste Unidade, Teste Sistema, Teste Funcional, Teste Estrutural, Caixa Branca, Caixa Preta, Qualidade de Software.

Abstract

Vieira Pessoni, Vinícius. **Software Testing Proccess Implantation**. Goiânia, 2011. 61p. Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

This document aims bringing to light questions about how the test effort can contribute to quality in software based systems. Therefore, this project is an study case applied to CERCOMP-UFG instituition wich proposes a construction of a knowledge body about software testing, a testing proccess and templates according with it's needs.

Keywords

Software Testing, CERCOMP, Testing Process, Component Test, System Test, Black Box Test, White Box Test, Software Quality.

Glossário

1. Anomalia: qualquer sintoma, comportamento ou resultado diferente do esperado ou previsto.
2. Ciclo de Teste: Execução da massa de teste sobre o produto de software a cada versão que é liberada. Um passo pode conter um ou mais ciclos de teste.
3. Critério de Cobertura: de maneira simplificada é o que define regras práticas para quando parar os testes.
4. Defeito/Bug: passo, processo ou definição de dados incorretos. Ou ainda qualquer problema existente no software ou sistema sob teste que pode causar sua falha em atender as expectativas do usuário. Em outras palavras, um defeito é uma fonte potencial de insatisfação com o produto.
5. Engano: ação humana que produz um defeito.
6. Erro: estado inconsistente, observado em momento de execução, de um produto ou sistema de software. Esse erro pode ou não causar uma falha.
7. Falha: estado inseperado do produto ou sistema de software ocasionado por um erro.
8. Ferramenta de teste: qualquer ferramenta de hardware e/ou software de propósito geral utilizada durante a execução de um caso de teste. Podendo ser utilizada para configurar o ambiente, criar condições de teste, ou medir os resultados dos testes. A ferramenta de teste geralmente é separada do caso de teste em si.
9. Feedback: informações do ponto de vista de usuário sobre determinado produto, atividade, ou assunto. Pode ainda ser entendido como a avaliação de determinado assunto.
10. Gerente de Teste: deve garantir que os testadores identifiquem, reproduzam e isolem os defeitos. É ainda parte do seu trabalho retestar, rastrear defeitos até suas conclusões e prover relatórios desses defeitos aos gerentes e executivos. OBS: A responsabilidade da gerência de correção dos defeitos pertence ao gerente de desenvolvimento, e não ao gerente de teste. Ou seja, é o gerente de desenvolvimento quem deve definir quais defeitos serão corrigidos, qual desenvolvedor irá corrigir qual defeito e quando ele irá fazer isso.
11. Passo de teste: tempo necessário para executar todos os testes, de todos os conjuntos

de teste uma vez.

12. Testadores: responsáveis por configurar o ambiente de teste, iniciar os testes, interpretar resultados, reproduzir anomalias manualmente, isolar defeitos por experimentação, e restaurar o ambiente ao seu estado anterior à configuração para execução do teste. Em adicional possuem a responsabilidade de dar manutenção no sistema de teste, especialmente nos casos e conjuntos de teste.
13. Unidade, Componente: menor parte de um software, definido quando utilizado. Exemplo: função, método, classe.

Sumário

Lista de Figuras

CAPÍTULO 1

Introdução

O processo de software define diretivas para que as equipes de desenvolvimento possam realizar com sucesso as etapas da produção de software. Aliado a esse processo, preferencialmente em paralelo, deve existir um processo de teste, o qual possui a responsabilidade de auferir a qualidade tanto do processo quanto do produto em desenvolvimento.

O assunto tratado por este projeto é a contribuição que a atividade de teste pode trazer no quesito qualidade dos produtos e processos de software. Nesse sentido, o projeto é um estudo de caso voltado a instituição CERCOMP-UFG [1] o qual conta com a construção de uma base de conhecimento para teste bem como a proposta de processos e roteiros adequados à essa atividade específicos para tal.

Os benefícios esperados com a implantação do processo de teste incluem maior qualidade dos softwares produzidos pelo CERCOMP, além de redução do tempo gasto com manutenções corretivas, bem como a possibilidade de visualização de métricas antes não coletadas em tempo de desenvolvimento para medir também a qualidade dos processos de desenvolvimento.

Este documento possui sete capítulos, os quais encontram-se organizados da seguinte maneira: Introdução, capítulo atual, com breve explicação sobre o assunto e visão geral do documento; Fundamentação Teórica, capítulo contendo os conceitos chave dos processos, metodologias e documentações apresentadas; A instituição, capítulo em que é apresentado um panorama da atual situação do CERCOMP; Base de conhecimento, capítulo no qual são descritos os componentes para a base de conhecimento de teste; Processo de teste, capítulo que contém o processo proposto, bem como as explicações sobre esse; Relacões de Entradas e Saídas para cada Tarefa de Teste, capítulo no qual são descritos de forma tabular as entradas e saídas para cada fase do processo de teste; Conclusão, capítulo que apresenta a conclusão do projeto, e propõe os trabalhos futuros.

[1] Centro de Recursos Computacionais da Universidade Federal de Goiás - O CERCOMP está ligado à Pró-reitoria de Desenvolvimento Institucional e Recursos Humanos e é composto de uma diretoria, uma secretaria, um serviço de atendimento ao usuário (SAU), três divisões de produção (a saber: Divisão de Sistemas, Divisão de Redes e Divisão de Suporte e Treinamento), um grupo de comissões técnicas e um conselho técnico. http://www.cercomp.ufg.br/

CAPÍTULO 2

Fundamentação Teórica

Nesta seção são apresentados as informações fundamentais que formam as bases para o bom entendimento da atividade de teste de software. Também é apresentada uma visão geral sobre essa atividade passando por suas fases, técnicas, requisitos, critérios e politicas de teste. Essa visão é enriquecida com detalhes no capítulo 5 que contém o processo de teste proposto a instituição. Os conceitos explanados nestas subseções serão utilizados por todo o documento.

2.1 Software

"Software não é apenas o programa, mas também toda a documentação associada e os dados de configuração necessários para fazer com que esse programa opere corretamente. Um sistema de software, usualmente, consiste em uma série de programas separados, arquivos de configuração que são utilizados para configurar esses programas, documentação de sistema, que descreve a estrutura desse sistema, e documentação de usuário, que explica como utilizar o sistema" [12].

"Software são instruções (programas de computadores) que, quando executadas, fornecem características, função e desempenho desejados; e documentos que descrevem a operação e uso dos programas" [11].

As duas definições anteriores sobre o conceito de software são provenientes de dois dos mais conhecidos autores de textos sobre a Engenharia de Software. É importante definir bem esse conceito, pois nesse documento são trabalhados diretamente tanto com os produtos de software, quanto com os processos que dão origem a tais produtos.

Essas definições pregam que software é muito mais do que apenas as linhas de código, ou os programas somente. O conceito de software é muito mais amplo, abarcando todo o conhecimento por trás de sua construção e utilização. Dessa forma, quando se trata de qualidade, é preciso enxergar de maneira grandiosa, e pensar na obtenção da qualidade como um todo, partindo desde a menor unidade do software, até o nível de sistema.

2.2 Qualidade de Software

Embora qualidade seja um conceito relativo, e em certas vias pessoal, no contexto dos produtos de software a qualidade encontra-se intimamente ligada às funções que esses desempenham. Nesse sentido, cada tipo de software possui um conceito de qualidade. Por exemplo, um software de banco deve ser seguro, um tocador de mídia deve reproduzir com coesão os formatos a que se presta reproduzir, um editor de texto deve ser capaz de editar e realizar as mudanças que o usuário necessita sobre textos, e assim por diante cada qual em seu contexto.

Apesar das diferentes finalidades, softwares ditos de qualidade possuem atributos que são comuns a tais e que podem ser observados. Esses atributos servem como métrica dessa qualidade e no padrão ISO 9126-1 4.1.1 são classificados nas seguintes características:

- Funcionalidade
- Portabilidade
- Confiabilidade
- Manutenibilidade
- Usabilidade
- Eficiência

Hoje existe uma norma de qualidade mais atual, a família de normas ISO 25.000.
A iso 25.010, componente dessa família, descreve um modelo de qualidade com novas características.
Essa norma "substituiu" a iso 9126, englobando partes delas.

Qualidade de software visa então a garantir, primordialmente, a conformidade do produto com os requisitos obtidos para a sua produção. Um bom processo de software, com bons programadores e planejamento inteligente, contribui, mas não é suficiente para garantir um software correto e sem defeitos, ou seja, um produto de boa qualidade. E nessa busca pela qualidade encontram-se maneiras de investigação em diferentes níveis sobre essa conformidade, sendo uma delas a atividade de teste de software desenvolvida nesse trabalho.

2.3 Verificação e Validação(V&V)

Verificação e Validação, apesar de estarem intimamente ligadas, não são a mesma coisa. Os processos de verificação e validação (V&V) determinam quando os produtos desenvolvidos para uma determinada atividade estão em conformidade com os requisitos daquela atividade. Em termos de software, pode ser entendido como o quanto o software satisfaz os requisitos e as necessidades do usuário.

Os requisitos de processo do ciclo de vida de verificação e validação são especificados para diferentes níveis de integridade. O escopo dos processos de V&V

abarca sistemas baseados em software, software de computador, hardware e interfaces. Assim processos de V&V incluem análise, avaliação, revisão, inspeção, graduação e teste de produtos de software.

O propósito da V&V de software é auxiliar as organizações a acrescentar qualidade ao produto durante seu ciclo vida de desenvolvimento. Adicionado a isso, os processos de V&V proporcionam uma avaliação objetiva dos produtos de software, durante todo esse ciclo. Por meio dessa avaliação é possível demonstrar o quanto os requisitos de software e de sistema são corretos, completos, coerentes, consistentes e testados.

Os processos de V&V podem ser enxergados como uma extensão do gerenciamento de projetos e engenharia de sistemas que emprega uma metodologia rigorosa para identificar dados objetivos e conclusões. A partir disso, é possível prover um *feedback* sobre a qualidade do software, desempenho, e andamento da agenda para os interessados.

Esse *feedback* consiste de informações como resultados fora do grupo dos esperados, melhorias de desempenho e qualidade, não apenas para as condições operacionais previstas, mas também a nível de sistemas e suas interfaces.

Os resultados de *feedback's* anteriores permitem que os desenvolvedores possam comparar a evolução do software, e possam mudá-lo em tempo hábil de desenvolvimento, reduzindo assim o impacto no projeto como um todo. Se não houver uma atitude proativa, anomalias e mudanças associadas ao sistema de software são normalmente procrastinadas o que resulta em maior custo de produção do software, e ainda em maior custo acarretado pelo atraso no cronograma.

Os processos de V&V são divididos em: processo de verificação e processo de validação tradados em mais detalhes a seguir.

O processo de verificação tem como objetivo evidenciar quando o software, seus produtos e processos associados:

- Estão em conformidade com os requisitos funcionais e não funcionais: isso é feito verificando-se a correção, completude, consistência, coerência etc. Essa verificação é realizada durante todas as atividades do ciclo de vida, durante cada processo do ciclo de vida (aquisição, fornecimento, desenvolvimento, operação e manutenção).
- Satisfazem os padrões, práticas e convenções durante o processo do ciclo de vida.
- Completam com sucesso cada atividade do ciclo de vida e satisfaz todos os critérios para prosseguir para a próxima fase do ciclo.

O processo de validação tem como objetivo evidenciar quando o software, seus produtos e processos associados:

- Satisfazem os requisitos de sistema determinados para eles. Essa validação é realizada ao final de cada atividade do ciclo de vida.

- Solucionam o problema correto, ou seja, se as implementações das regras de negócio estão corretas.
- Satisfaz o uso pretendido e as necessidades do usuário.

Como enunciado anteriormente, os processos de verificação e validação estão interligados. Nesse sentido, esses processos são complementares e utilizam os resultados um do outro para estabelecer melhores critérios de análise, verificação, revisão, avaliação, inspeção e tarefas de V&V para cada atividade do ciclo de vida do software.

O padrão ISO/IEC 12207 4.1.2 prega que a equipe de desenvolvimento deve realizar vários testes e avaliações como parte integrante do processo de desenvolvimento. Assim as técnicas descritas por esse padrão são úteis no sentido de definir em detalhes os passos para a realização desses testes e avaliações.

2.4 Teste de Software

O teste de software se trata de uma técnica dinâmica de verificação e validação. Assim teste de software pode ser visto como a atividade de executar a aplicação com determinadas entradas, denominados dados de teste, e examinar as saídas juntamente com seu comportamento para saber se ele está realizando o que se espera.

Uma definição alternativa sobre teste é “Teste é um processo de executar um programa com a intenção de fazê-lo falhar”. É preciso tomar cuidado para não ser tendencioso ao testar um software, pois tendo essa filosofia em mente, as chances de encontrar erros são maiores. Pode-se então entender o processo de teste como “um processo destrutivo no qual tentamos encontrar defeitos, os quais a existência é tomada como premissa”[10].

O objetivo do teste de software é auxiliar a instituição desenvolvedora a acrescentar qualidade ao produto de software durante o seu ciclo de desenvolvimento e validar que determinado grau de qualidade foi alcançado.

É importante ressaltar que o teste de software não tem por objetivo, como é de senso comum, mostrar que o produto está correto. Ao contrário do senso comum, o objetivo é fazer o produto falhar, e encontrar os defeitos desse produto, o que reforça a natureza destrutiva do teste. Com a demonstração dos defeitos e a identificação das falhas é possível então direcionar o trabalho de correção da equipe de desenvolvimento.

Outra finalidade do teste de software é garantir que os sistemas de software funcionam como se espera quando utilizados pelos seus usuários. Sendo assim, a atividade de teste procura por situações nas quais o produto falha em atender as expectativas de um consumidor ou usuário em determinada área.

Uma estratégia geral de teste pode ser definida contendo as seguintes etapas:

- Planejamento
- Execução do teste
- Análise dos Resultados

Essas três etapas gerais de teste estarão presentes em qualquer estratégia de teste que seja bem estruturada. Em algumas literaturas [6] a etapa de planejamento é dividida em duas etapas que correspondem a etapa de planejamento e etapa de projeto dos casos de uso. A abordagem supracitada diz respeito ao modelo geral, e o processo proposto para a instituição no capítulo 5 faz uso dessa segunda divisão, tendo portanto quatro etapas ao invés de três. Essa diferença na quantidade de etapas diz respeito apenas a sistematização dos passos a serem executados. Dessa forma o processo com três e quatro divisões são equivalentes em conteúdo.

É comum a algumas instituições deixarem a atividade de teste somente para o final do processo de desenvolvimento. Essa abordagem é extremamente problemática pois na maioria das vezes ao final desse processo dificilmente haverá tempo, dinheiro e pessoal o suficiente para que essa atividade seja desempenhada com a devida atenção. Apesar de não ser impossível realizar os testes somente ao final do desenvolvimento, o correto, segundo [5, 6, 10] é que a atividade de teste seja executada em paralelo ao processo de desenvolvimento, de modo que para cada etapa do desenvolvimento haja um correspondente no processo de teste.

Ao se integrar o processo de desenvolvimento a atividade de teste em todos os estágios do ciclo de vida do processo de desenvolvimento pode-se aumentar dramaticamente a efetividade e a eficiência do teste. O principal argumento dessa integração é o ganho expressivo na qualidade dos softwares desenvolvidos.

Sendo assim a seguinte premissa deve ser adotada, quando o assunto é teste de software: quanto mais cedo um defeito for identificado melhor e mais barato ele se torna para ser corrigido, e reciprocamente, quanto mais tardar ao identificá-lo, mais irá impactar no produto, e mais caro será para corrigi-lo.

2.5 Plano de Teste

Plano de teste está relacionado com a descrição dos produtos de teste e o estabelecimento de padrões para o processo de teste. Esse também possui a função de ajudar os gerentes de teste na alocação de recursos e estimativa de agendas de teste. Assim esses planos auxiliam a equipe técnica a ter uma visão geral do sistema e da execução dos testes.

Além de definir a agenda de teste e os procedimentos necessários para testar o produto, o plano de teste define a infraestrutura necessária para tais. Nesse sentido, no

plano de teste encontram-se também as anotações sobre os recursos humanos, de hardware e software necessários. Essas estimativas são uteis principalmente para os gerentes de projetos e configuração que são responsáveis por garantir que esses recursos estarão disponíveis para a execução dos testes.

O plano de teste, segundo Sommerville [12], é composto das seguintes informações:

- Processo de teste: uma descrição das principais fases do processo de teste.
- Rastreabilidade de Requisitos: o interesse maior dos usuários é que o sistema esteja conforme suas necessidades e o teste deve ser planejado de modo que cada requisito seja individualmente testado.
- Itens testados: os produtos do processo de software que serão testados devem ser identificados.
- Agenda de teste: uma agenda geral e a alocação de recursos para cumprir essa agenda em sincronia com a agenda do projeto de desenvolvimento.
- Registro dos procedimentos de teste: executar os testes não é suficiente. Os resultados desses testes devem ser sistematicamente anotados. Deve ser possível a auditoria do processo de teste para que seja verificado se esse foi executado corretamente.
- Restrições: restrições que afetam o processo de teste tais como diminuição da equipe, devem ser previstas nessa seção.

Para sistemas pequenos, um plano menos formal pode ser utilizado, mas ainda existe a necessidade de um documento formal para auxiliar o planejamento do processo de teste. Os planos de teste não são documentos estáticos, pois esses evoluem devido a adversidades que podem ocorrer durante o processo de desenvolvimento.

A padronização dos planos de teste está descrita na norma IEEE 829 4.1.4 a qual está presente na seção que compõe o corpo de conhecimento juntamente com uma breve explicação sobre ela. Mais informações e detalhamentos sobre plano de teste encontram-se na seção 5.1.1.

2.6 Casos de Teste

Casos de teste em um nível mais alto, podem ser pensados como uma sequência de ações, onde cada ação pode possuir dados necessários a esse teste e resultados associados. De maneira simplificada, casos de teste são especificações de conjuntos de entradas a serem testadas juntamente com as saídas esperadas, adicionando se ainda, informações sobre o que está sendo testado.

Uma outra definição, pensando-se em ações pode ser utilizada para casos de teste. Sendo assim um caso de teste é onde as ações são executadas sobre o sistema que está sendo testado. Isso quer dizer que os dados são inseridos e trabalhados pelo sistema de teste, e o sistema de teste é levado a algum estado ou estados diferentes ou iguais aos iniciais.

O conjunto de três componentes - ação, dados e resultados esperados - é onde o teste acontece e as condições de teste são criadas. Assim, de posse dos resultados de cada caso de teste e comportamentos desses, os testadores podem realizar uma comparação com os resultados esperados.

O princípio fundamental dos casos de teste é que esses devem ser escritos de maneira metódica e bem estruturada para que sejam repetíveis/reutilizáveis. Isso é necessário para o caso da confirmação de uma anomalia do sistema, no qual se executa um caso de teste por repetidas vezes até que sejam encontrados os defeitos dos quais se suspeita a existência.

A geração automatizada dos casos de teste é possível para domínios muito pequenos, mas inviável para grandes domínios. A saída dos casos de teste deve ser interpretada por pessoal que entende o que o sistema deveria fazer, ou ainda algumas ferramentas que automatizam essa análise, os quais são conhecidos como oráculos.

2.6.1 Como escrevê-los

O ponto mais importante na atividade de teste é o conhecimento de como escrever casos de teste efetivos, ou seja, os que encontrem a maior quantidade de defeitos possível. Em um teste ideal, todos os caminhos possíveis de um determinado software deveriam ser executados. No entanto testar todos os caminhos, mesmo em um software pequeno não é trivial devido à infinidade de combinações de entradas e saídas que ele pode oferecer. Assim a criação de casos de teste para todos os possíveis casos é impraticável.

Se para aplicações simples, o teste completo é impraticável, para aplicações complexas o teste completo é praticamente impossível, pois demandaria enormes quantidades de tempo e recursos - tanto humanos quanto financeiros - para realizá-lo. Essa exorbitante demanda seria totalmente inviável e é exatamente por isso que o desenvolvimento de casos de teste é tão importante.

Assim como a cobertura de todos os possíveis caminhos é impraticável, e na maioria das vezes impossível, encontrar todos os defeitos em um programa também o é. Observando-se as limitações de custo, pessoal e tempo, um bom conjunto de teste é aquele composto por bons casos de teste, ou seja, os que possibilitam a identificação da maior quantidade de defeitos possível.

2.7 Técnicas de teste

2.7.1 Teste Funcional (Caixa Preta)

Também referenciado como teste executado na interface, ou ainda teste comportamental o teste funcional é uma técnica utilizada para se projetar casos de teste cujo o foco é o comportamento da aplicação. Isso é feito fornecendo-se entradas ao software e analisando a coerência das saídas produzidas por esse. O termo "caixa preta" é lhe dado por analisar o software da ótica de uma caixa preta, podendo ainda ser pensado em termos de "o que o software deve fazer".

Nessa técnica os detalhes de implementação não são considerados e o software é avaliado segundo o ponto de vista do usuário. Por esse motivo, critérios dessa técnica podem ser aplicados em todas as fases de teste - unidade, integração, sistema - em produtos desenvolvidos em diferentes linguagens e paradigmas de programação.

A realização do teste funcional envolve alto conhecimento do domínio da aplicação e da problemática do negócio ao qual ele servirá. Dessa forma, testes funcionais são melhor realizados por testadores que entendem do design do sistema, pelo menos em alto nível. As informações necessárias para a construção dos casos de teste nesse nível provem principalmente do documento de requisitos e da documentação do software. Os critérios mais conhecidos do teste funcional são:

- Particionamento em Classes de Equivalência
- Análise do Valor Limite

O processo proposto recomenda o uso desses dois critérios para a geração de casos de teste tanto nas fases de unidade quanto de sistema por meio da técnica funcional. Uma limitação séria dos critérios de testes funcionais é a impossibilidade de assegurar que partes essenciais ou críticas do produto em teste foram executadas com o conjunto de teste. Para contornar essa impossibilidade é que se utilizam os critérios estruturais, como forma de se medir quais partes do código foram ou não avaliadas. No processo proposto essa complementação é recomendada.

2.7.2 Teste Estrutural (Caixa Branca)

Certas vezes referenciado por caixa de vidro, essa técnica requer que os caminhos lógicos do software sejam testados. Os casos de teste dessa técnica são focados em verificar conjuntos específicos de condições, estruturas de seleção, repetições, definições e utilização de variáveis. Os critérios pertencentes à técnica estrutural são classificados com base na complexidade, no fluxo de controle e no fluxo de dados [8].

Nesse contexto, teste estrutural envolve um conhecimento detalhado do sistema pois é feito olhando-se o código fonte, estruturas de dados, bem como outros detalhes de implementação. Dessa forma os testadores desse nível são geralmente os desenvolvedores pois possuem habilidades sobre esse conhecimento.

Os critérios estruturais permitem identificar quais partes do produto em teste foram ou não executadas pelo conjunto de teste, sendo dessa maneira, complementares aos critérios funcionais. Assim sendo, no contexto desse trabalho, os critérios estruturais serão utilizados como métrica de qualidade dos conjuntos de teste funcionais construídos.

Os critérios estruturais mais comuns são:

- Teste de caminho básico
- Teste de condição
- Teste de fluxo de dados
- Teste de ciclo (loops)

Para o processo proposto nesse documento o critério todos os nós ou todos os comandos é recomendado para a geração dos casos de teste por meio da técnica estrutural. Esse critério é utilizado somente na fase de unidade não sendo portanto utilizado na fase de sistema.

2.8 Fases da atividade de teste (Níveis de Teste)

Assim como o processo de desenvolvimento, a atividade de teste é segmentada em fases de forma a reduzir a complexidade e possibilitar a atenção dos testadores para aspectos específicos em cada uma delas. Dessa maneira, a atividade de teste apresenta uma divisão que compreende três fases principais com objetivos distintos, sendo elas: fase de unidade, integração e sistema. Essas fases estão delineadas na Figura 2.1 juntamente com sugestões de técnicas a serem aplicadas em cada uma delas, e são descritas nas subseções seguintes.

2.8.1 Teste de Unidade

Fase que busca testar a menor unidade de um programa. No paradigma estruturado essa unidade pode ser entendida como uma função. No paradigma orientado a objetos pode ser um método ou uma classe, dependendo do contexto.

Nessa fase realiza-se a busca por defeitos algorítmicos, lógicos, estruturas de dados errôneas, ou mesmo enganos de programação. Em virtude da necessidade de ser executada em fase de desenvolvimento, essa fase é comumente realizada pelo próprio desenvolvedor da unidade. O processo proposto prevê a realização de testes nessa fase.

2.8.2 Teste de Integração

Visa testar a interação entre as unidades do software, investigando as relações e trocas de informações entre essas. Essa fase também exige grande conhecimento das estruturas internas das unidades, e de como elas se relacionam, sendo assim, também costuma ser executada pelos desenvolvedores. O processo proposto não prevê a realização de testes nessa fase devido a maturidade da instituição.

2.8.3 Teste de Sistema

Fase realizada após as unidades serem testadas e integradas. Tem por objetivo a busca de defeitos em nível de sistema, ou seja, avalia se as funcionalidades do sistemas estão de acordo com os requisitos. Além da verificação de conformidade com os requisitos funcionais são analisados os requisitos não funcionais como segurança, desempenho, robustez dentre outros. Essa fase, em geral costuma ser executada por uma equipe alheia a equipe de desenvolvimento. O processo proposto prevê a realização de testes nessa fase.

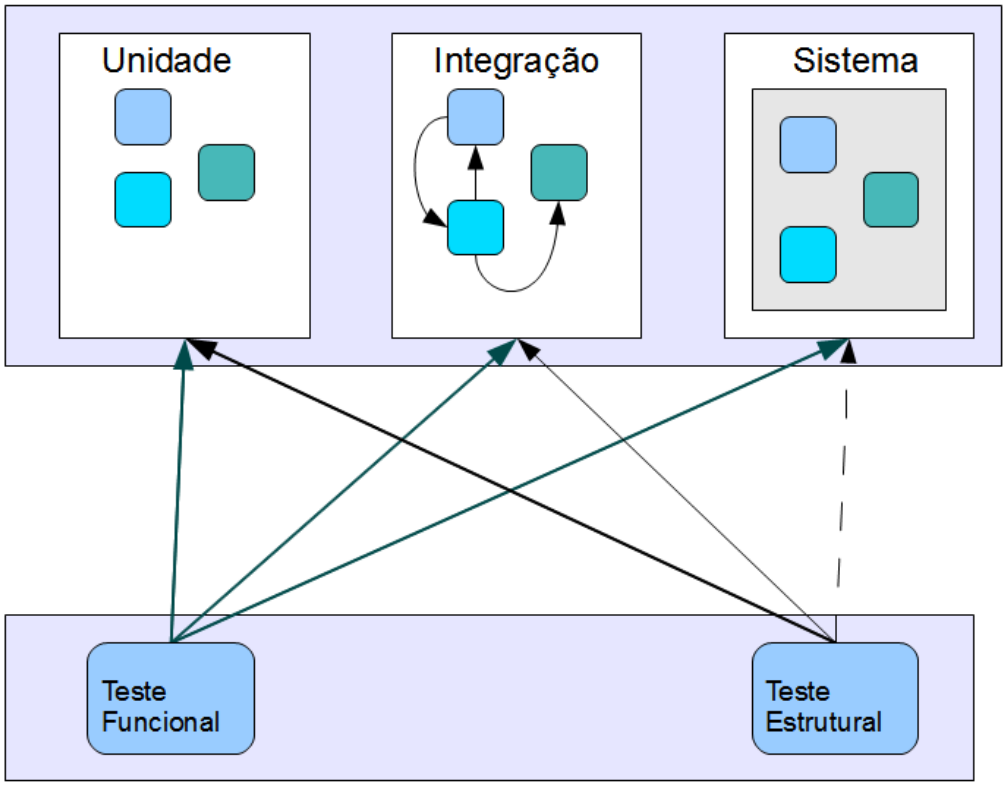

Figura 2.1: *Fases de Teste relacionadas as Técnicas de Teste*

2.9 Sistema de Rastreamento de Defeitos

Também conhecido e referenciado como bug em algumas literaturas, cada defeito encontrado por um caso de teste deve ser registrado num sistema de armazenamento de defeitos nomeado por um identificador único e inequívoco, compondo uma base de dados sobre os defeitos encontrados.

Dessa forma, um sistema de rastreamento de defeitos é um programa ou aplicação que permite que a equipe de teste possa armazenar, gerenciar, e analisar os relatórios de defeitos. Esse sistema manterá esses registros de forma que seja possível inserção, remoção e busca desses defeitos, facilitando a organização das informações sobre o que foi encontrado e o que já foi corrigido.

O ideal é que o sistema de rastreamento de defeitos seja um sistema de banco de dados devido a quantidade de informações para lidar e devido a facilidade com que se pode extrair métricas e gráficos de uma dessas aplicações. Atualmente existe uma série de sistemas de rastreamento de defeitos, tais como o BugZilla [1] ou o Mantis [2], mas caso haja dificuldade em instalar ou gerenciar um desses sistemas, uma simples planilha pode ser empregada para fazer o rastreamento dos defeitos encontrados.

2.10 Requisitos de Teste

Também conhecido como elemento requerido, segundo Offut [5] requisito de teste é um elemento específico de um artefato de software que um caso de teste deve satisfazer ou cobrir. Esses requisitos de teste podem ser entendidos como as combinações das possibilidades de entrada para um determinado produto, ou ainda, as condições a serem testadas.

Os requisitos de teste podem ser obtidos de várias fontes de informações, sendo as mais comuns o documento de requisitos do produto em teste e o código fonte desse. São esses requisitos de teste que guiarão a construção dos casos de teste. Assim, deve ser criado um caso de teste para cada um desses requisitos. Importante notar que um caso de teste pode cobrir mais de um requisito de teste, eventualmente.

[1] http://www.bugzilla.org/
[2] http://www.mantisbt.org/

2.11 Critérios de Cobertura de Teste

Segundo Offut [5] critérios de cobertura são, de forma simplificada, regras para gerar requisitos de teste por meio de uma forma sistematizada. Dessa maneira esses critérios de cobertura de teste podem ser vistos como os meios pelos quais os requisitos de teste são gerados.

Cada técnica de teste - estrutural, funcional - possui critérios específicos, sendo que alguns desses critérios podem ser comuns às duas técnicas, a exemplo, o particionamento de equivalência. A seguir são definidos os critérios utilizados nesse documento para cada técnica:

- **Estrutural:**
  - particionamento em classes de equivalência ;
  - análise do valor limite.
- **Funcional:**
  - particionamento em classes de equivalência;
  - todos os caminhos ou todos os comandos.

Os critérios de cobertura estão intimamente ligados aos casos de teste, no sentido que esses derivam requisitos de teste, os quais por sua vez servem como guias de como os casos de teste devem ser criados. Um esquema desse relacionamento está definido na Figura 2.2.

2.12 Sistema de Teste

Sistema de teste pode ser entendido como a capacidade, criada a partir de um conjunto de fatores, de se testar um produto de software. Os três fatores que proporcionam essa capacidade são:

- o processo de teste: abarca tanto os procedimentos escritos quanto os não escritos e definições sobre como uma equipe realizará seus testes;
- o testware: inclui todas as ferramentas, documentações, dados, casos de teste, mecanismos de rastreamento de defeitos, e assim por diante, utilizados pela equipe de teste;
- o ambiente de teste: inclui laboratório, hardware, software, redes e toda infraestrutura necessária para que a equipe de teste possa realizar com sucesso a atividade de teste.

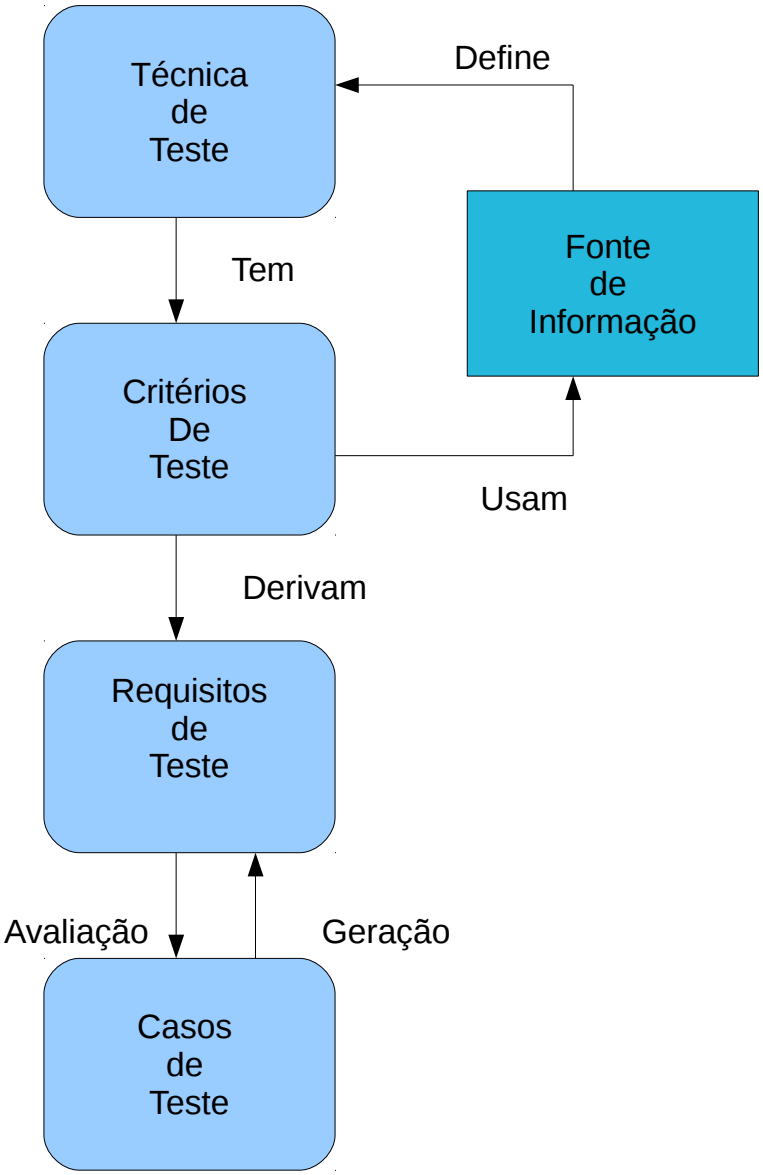

Figura 2.2: *Relacionamento entre Técnica, Critérios, Requisitos e Casos de Teste.*

A qualidade de um sistema de teste pode ser avaliada de acordo com os atributos sugeridos pela NBR ISO/IEC 9126 citados na Seção 2.2 ou avaliada de acordo, com a análise de risco de qualidade. Porém a qualidade mais importante do sistema de teste é fornecer as funcionalidades necessárias para a execução dos testes.

O sistema de teste deve ser confiável. Isso se reflete no sentido de que quando submetido as mesmas entradas em mesmas condições deve produzir as mesmas saídas. Observe que essa condição é válida apenas para programas determinísticos. No caso de programas não determinísticos é possível que para uma mesma entrada o programa apresente resultados corretos diferentes.

Esse sistema deve ainda permitir a execução de casos de teste de maneira fracamente acoplada. Em outras palavras, em posse de dois casos de teste CTa e CTb sem relação de dependência entre eles, ao executá-los ao mesmo tempo, separados, ou em qualquer ordem o resultado de um - falha ou sucesso - não deve influenciar no resultado do outro.

Um bom sistema de teste também deve ser eficiente, portável, manutenível e flexível. A manutenibilidade é extremamente importante pois o próprio desenvolvimento é altamente mutável. Isso quer dizer que os casos de teste deverão sofrer manutenção todas as vezes que o comportamento do sistema for modificado de alguma forma, ou mesmo quando se deseja que o critério de cobertura de teste seja modificado.

Assim como um sistema de software, um sistema de teste deve possuir documentação que descreva sua arquitetura, como ele funciona e como utilizá-lo. Isso é alcançado com nomes adequados e intuitivos para seus componentes e relativa simplicidade em sua implementação. Em suma o sistema de teste existe para dar suporte a execução dos casos de teste no sistema em teste.

2.13 Definição da Política de Teste

Definir quais as fases do processo de desenvolvimento o processo de teste irá abranger é uma decisão singular de cada instituição. Isso significa que é ela quem define se serão executados teste de unidade, integração e sistema sucessivamente. Essa instituição pode, por exemplo, optar por não executar teste de integração e fazer somente teste de unidade, sistema e/ou regressão. Ou ainda, dependendo da magnitude e abrangência do software, somente teste de sistema.

Vários são os pontos que devem ser analisados para definir uma política de teste, alguns deles podem ser: maturidade da instituição, custo das etapas, criticidade do software, pessoal e tempo disponíveis. Outros quesitos podem ser levados em consideração nessa definição, mas fica a cargo da instituição defini-los.

O processo proposto nesse trabalho sugere que sejam realizados testes nas fases de unidade e sistema, não cobrindo portanto as fases de integração nem regressão. Essa decisão é deliberada e motivada pelas necessidades e maturidade da instituição.

CAPÍTULO 3

A Instituição

O Centro de Recursos Computacionais da Universidade Federal de Goiás - CERCOMP está ligado à Pró-reitoria de Desenvolvimento Institucional e Recursos Humanos e é composto de uma diretoria, uma secretaria, um serviço de atendimento ao usuário (SAU), três divisões de produção (a saber: Divisão de Sistemas, Divisão de Redes e Divisão de Suporte e Treinamento), um grupo de comissões técnicas e um conselho técnico [1].

Essa instituição é responsável pelo desenvolvimento dos sistemas utilizados no âmbito da universidade. É de sua competência sistemas relacionados ao controle, gerência, matrículas bem como os sites informativos, dentre outros sistemas de software.

3.1 Situação Atual da Instituição no Contexto de Desenvolvimento

A instituição atualmente possui as seguintes divisões internas que trabalham diretamente com o desenvolvimento de produtos de software:

- Divisão de Sistemas a qual é subdividida em:
  - Sistemas 1
  - Sistemas 2
- Equipe Web

A instituição também possui um grupo de processo de software (GPS) responsável pela elaboração, manutenção e implantação dos processos de software desenvolvidos para tal.

Além disso, existe ainda um processo de construção de software bem estruturado o qual está descrito em seu "Manual de Produção de Software do Cercomp". Esse manual

[1] http://www.cercomp.ufg.br/

é relativo a segunda versão desenvolvida pelo GPS da instituição e conta com a descrição detalhada de cada etapa do processo com modelos a serem seguidos para se obter cada artefato.

Apesar da satisfatória estruturação do processo, e de bons modelos desenvolvidos, é ausente nesse processo a descrição de um processo específico de teste que se enquadre nas necessidades de cada área. Nesse sentido, esse trabalho visa a proposição de um processo de teste que seja adequado a essas necessidades.

Atualmente várias tecnologias são trabalhadas nas divsões citadas anteriormente pela instituição, sendo algumas delas: Oracle Forms, HTML, PHP, RUBY, dentre várias outras. Apesar dessa variedade de tecnologias duas delas possuem o uso mais difundido dentre os sistemas existentes. As tecnologias mais utilizadas nos sistemas desenvolvidos são: Oracle Forms (consequentemente JAVA pois sua versão mais atual é executada na JAVA EE [2]) e PHP portanto demandaremos maior atenção para tais tecnologias.

O foco do projeto são os produtos de software desenvolvidos pelas divisões de Sistemas 1 e Sistemas 2. A relação dos sistemas existentes nessas divisões, juntamente com as tecnologias empregadas em cada um deles encontram se na seção "Relação de Sistemas" 3.2.

3.2 Relação de Sistemas

A tabela com a relação dos sistemas e as respectivas tecnologias empregadas em cada um deles encontra-se no ANEXO 1. O resumo das informações encontra-se no gráfico da Figura 3.1.

A maior dificuldade enfrentada na definição de um processo de teste que possa atender os vários sistemas existentes é provinda da heterogeneidade entre eles, ou seja, da variedade de linguagens, paradigmas e projetos arquiteturais desses sistemas.

Além das diferenças de tecnologias e de caráter arquitetural, existe ainda o plano de mudança de tecnologias e a unificação dessas por meio de um framework em desenvolvimento pela instituição em colaboração com o MEC [3].

Esse framework utiliza a linguagem JAVA, engloba várias outras tecnologias existentes bem como outras frameworks. Pretende-se com isso padronizar todos os novos sistemas desenvolvidos pela instituição. Além disso, objetiva-se ainda a reestruturação dos sistemas já desenvolvidos, de modo que sejam compatíveis com o novo framework.

[2] Java Platform, Enterprise Edition(JAVA EE): http://download.oracle.com/javaee/
[3] Ministério da Educação do Brasil, site disponível: http://portal.mec.gov.br/index.php

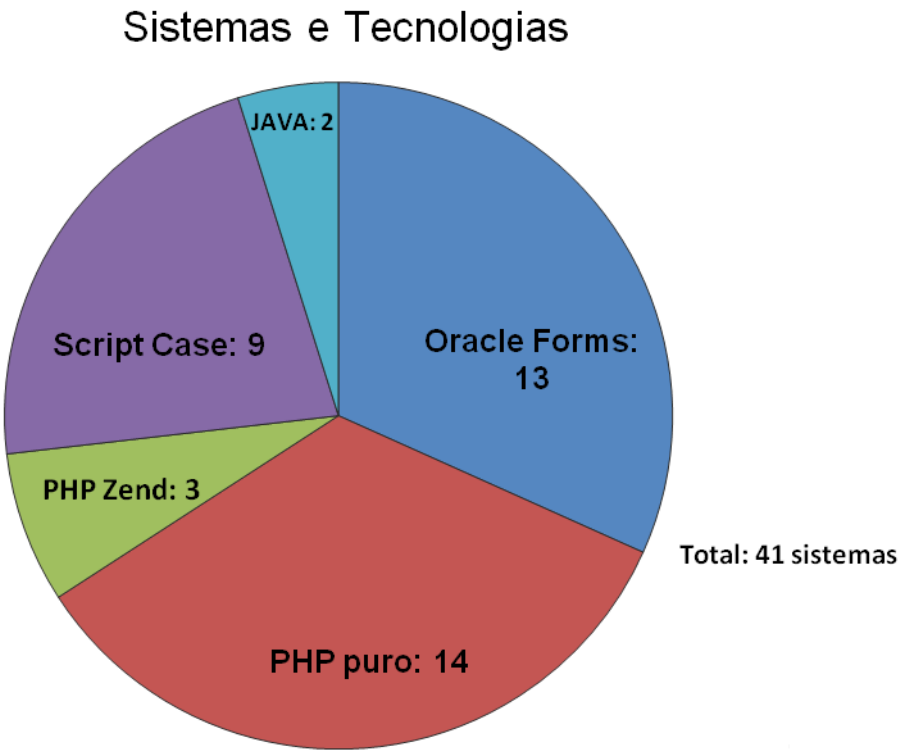

Figura 3.1: *Resumo de Sistemas por Tecnologia na Instituição.*

CAPÍTULO 4

Base de Conhecimento

Nesta seção estão descritas as recomendações de leituras relacionadas a atividade de teste de software. Essas recomendações farão parte de uma base de conhecimento a qual estará disponível na instituição de forma a auxiliar no aprendizado sobre a atividade de teste.

Nessas recomendações estão presentes normas e livros nos quais se encontram informações sobre como projetar, realizar e como documentar a atividade de teste. Nos subitens dessa seção, cada um desses materiais são comentados brevemente de forma a orientar o aprendizado utilizando esses recursos.

4.1 Relação de Normas Recomendadas

Normas são documentos publicados por organizações profissionais responsáveis por padronizar determinadas atividades, processos, dispositivos, etc. Nessas normas são descritas as etapas de processos, entradas e saídas esperadas de cada etapa, as formas e conteúdos de documentações, bem como outras informações necessárias a atividade de teste.

O motivo da existência de normas é para que se tenha uma padronização nas formas com que são realizados processos, produzidos documentos, e para que cada termo tenha o mesmo significado e seja entendido em diferentes situações facilitando a comunicação tanto entre as equipes internas de uma organização, quanto entre organizações diferentes.

A utilização de normas garante que todas as organizações que realizam determinada ação a faça de maneira similar, e que a comunicação entre essas organizações seja facilitada devido a essa similaridade. O uso de padrões também facilita a obtenção de um parâmetro de comparação para determinado atributo.

Nessa seção são explicadas normas provenientes de duas instituições:

- ISO: "International Organization for Standardization"ou em português, "Organização Internacional para Padronização".
- IEEE: "Institute of Electrical and Electronics Engineers", pronuncia-se I-3-e, ou em português "Instituto de Engenheiros Eletricistas e Eletrônicos".

As seguintes normas fundamentais para o processo de teste serão comentadas:

- ISO 9126-1 (2003)
- ISO 12207 (2008)
- IEEE 1012 (2004)
- IEEE 829 (2008)

4.1.1 ISO 9126 - 1 - Software Quality Characteristics

Esse padrão define um modelo de qualidade o qual pode ser aplicado a qualquer tipo de software sem fazer especificações sobre os requisitos desse produto. O objetivo desse padrão é proporcionar um framework para avaliação da qualidade dos produtos de software. Assim o modelo de qualidade definido por ele é baseado em seis características:

1. Funcionalidade: é a capacidade que o software tem de prover funções que atendam aos requisitos implícitos e explícitos. Pode ser entendido como “o que o software faz”.
2. Portabilidade: capacidade da transferência de um produto de software de um ambiente para outro.
3. Confiabilidade: capacidade que o produto de software tem de repetir sua funcionalidade dadas as mesmas condições antes aplicadas. Ou ainda, é a capacidade que um produto tem de executar determinada função sem sofrer desgaste ou envelhecimento.
4. Manutenibilidade: capacidade que o produto de software possui de ser modificado. As modificações incluem correções, melhorias ou adaptações.
5. Usabilidade: capacidade que o produto tem de ser compreendido pelo usuário. O quão fácil é para que o usuário aprenda e possa utilizar esse produto.
6. Eficiência: capacidade do produto de apresentar um desempenho satisfatório quando lhe é proporcionado recursos suficientes.

Cada uma dessas seis características pode ser detalhada em subcaracterísticas, as quais podem ser avaliadas direta ou indiretamente por meio de métricas de software.

4.1.2 ISO/IEC 12207 - Systems and Software Engineering - Software Life Cycle Processes

Essa norma descreve a arquitetura dos processos de ciclo de vida de software, sem especificar os detalhes de implementação ou execução das atividades e tarefas incluídas nos processos.

Esse padrão estabelece ainda um framework para processos de ciclo de vida de software, com terminologias bem definidas, que podem ser referenciadas por desenvolvedores de software. Nele são descritos processos, atividades e tarefas que são executadas durante a aquisição de um produto de software ou serviço, e durante o fornecimento, desenvolvimento, operação, manutenção e o descarte de produtos de software. Em determinados contextos software pode incluir porções de hardware.

Nesse padrão, as seções que terão mais importância no âmbito do teste são a Seção 6.4.6 na qual é discutido o processo de qualificação de teste de sistema e a Seção 7.1.7 referente ao processo de qualificação de teste de software.

Esse padrão cita ainda o padrão ISO/IEC 15288 que irá substituir o padrão ISO/IEC 12207 muito em breve. Assim é melhor investigar o novo padrão e basear as futuras modificações do processo de teste nesse novo padrão.

Como o novo padrão englobará as oito seções do padrão 12207 com algumas modificações, o padrão 12207 ainda é uma boa fonte de informações sobre os processos de ciclo de vida do software, e é utilizado como base para o desenvolvimento desse projeto.

4.1.3 IEEE 1012 - Standard for Software Verification and Validation

O IEEE Std 1012-2004 é um padrão de processo que define os processos de verificação e validação em termos de atividades específicas e tarefas relacionadas a cada um deles. O padrão também define os conteúdos do plano de V&V incluindo exemplos e formas os quais servem de diretivas.

Esse padrão se aplica ao desenvolvimento de software, manutenção e reuso. Os processos da V&V são definidos conforme o padrão ISO 12207 e referencia os processos de ciclo de vida do software, sendo compatível com todos os modelos de ciclo de vida. É importante notar, que nem todos os modelos de ciclo de vida utilizam todos os processos descritos nesse padrão.

Os objetivos desse padrão são:

- Estabelecer um framework para os processos de V&V, atividades e tarefas que contribuam de maneira direta para todos os processos de ciclo de vida de software. Inclusos nesses estão os processos de aquisição, fornecimento, desenvolvimento, operação e manutenção;

- Definir as tarefas de verificação e validação, conjunto de entradas e conjunto de saídas;
- Identificar as tarefas mínimas de V&V correspondentes a um esquema de integridade de 4 níveis;
- Definir o conteúdo de um plano de V&V.

Esse padrão pode ser inserido em todos os contextos de aplicações de software. Quando se conduz os processos de validação e verificação é importante examinar a relação que o software tem com o sistema do qual faz parte.

4.1.4 IEEE 829 - Standard for Software and System Test Documentation

A IEEE 829 provê um conjunto de padrões reconhecidos internacionalmente em documentação para planejamento de teste. Ela foi desenvolvida especialmente para esse fim e é aplicável a cada fase do ciclo de teste de software, incluindo teste de sistema e aceitação.

Esse padrão descreve um conjunto de documentos básicos, abarcando forma e conteúdo, que devem ser criados juntamente com a execução do processo de teste. A norma define a padronização da documentação de teste tanto em tempo de desenvolvimento, quanto as versões seguintes após completo esse desenvolvimento do produto de software.

Não são definidos nesse padrão metodologias específicas para se realizar o teste, bem como não são definidas técnicas, nem ferramentas para fazê-lo. A escolha das metodologias, técnicas e ferramentas cabe ao gerente de teste juntamente com a equipe de gerencia de projetos.

Alguns outros documentos de teste não descritos nessa norma podem ser necessários para a composição de uma boa documentação dependendo da metodologia utilizada, nesse sentido, fica a cargo do gerente de teste da instituição optar por quais desses documentos devem ser incluidos.

O padrão 829 é composto por oito documentos divididos em três áreas principais, sendo elas:

1. Plano de teste;
2. Especificação de Teste;
3. Relatórios.

Cada uma dessas seções são explicadas a seguir:

1. **Plano de teste:**

Descreve o escopo, estratégia, recursos e agenda das atividades de teste. Essa parte do documento também identifica os itens e funções a serem testadas, as tarefas de teste a serem realizadas, as pessoas responsáveis por essas, e os riscos associados ao plano.

2. **Especificação de Teste:**
   (a) Especificação do projeto de teste: refina as abordagens de teste identifica as funções a serem testadas. Além disso, deve identificar os casos de teste e os procedimentos de teste. Deve ainda especificar os critérios de aceitação/rejeição.
   (b) Especificação de Caso de Teste: a especificação do caso de teste abarca os valores de fato usados como entrada e suas respectivas saídas esperadas. Casos de teste são separados do plano de teste para possibilitar o uso em mais de um plano de teste e para permitir sua reutilização em diversas outras ocasiões.
   (c) Especificação de Procedimento de Teste: a especificação do procedimento de teste identifica todos os passos necessários para operar o sistema e exercitar um determinado caso de teste para que seja implementado determinado plano de teste. Procedimento de teste são separados do plano de teste pois eles não necessitam ser seguidos passo a passo nem conter muitos detalhes. Pode ser entendido com um apoio ao plano de teste, com informações adicionais sobre esse.

3. **Relatórios:**
   (a) Diário de teste: utilizado pela equipe para relatar o que ocorreu durante os testes.
   (b) Relatório de incidentes de teste: o relatório de incidentes de teste serve para descrever qualquer evento durante a execução do teste que necessita de mais atenção e verificação.
   (c) Relatório Resumo de teste: lista as atividades de teste de maneira associativa a um ou mais planos de teste.
   (d) Relatório de Encaminhamento de Item de teste: serve para identificar se os itens de teste são transmitidos corretamente entre as equipes de desenvolvimento e teste, haja vista que se presa para que sejam equipes diferentes

4.2 Relação de Livros Recomendados

Os livros recomendados para compor essa base de conhecimento são todos os presentes na seção de Referências Bibliográficas 6.1. Esses foram utilizados para desenvolver esse documento e para base do processo proposto.

CAPÍTULO 5

Processo de Teste

As normas descritas na seção 4.1, apesar de amplas, não são simples de serem inseridas no contexto do processo de desenvolvimento configurando a atividade de teste. Dessa forma, esse trabalho propõe maneiras práticas de utilizar as premissas dessas normas em prol da qualidade, por meio da definição de um processo de teste. O guia principal para as regras de documentação são provindas da IEEE 829 4.1.4.

O processo proposto possui foco na avaliação das funções que o software deve executar, ou seja, na validação e verificação de requisitos funcionais do software e suas restrições. O objetivo principal é garantir que o software se comporte conforme especificado no documento de requisitos, e de uma maneira razoável atenda as expectativas dos usuários.

A disposição do processo de teste em etapas possui a mesma motivação da segmentação do processo de desenvolvimento, e está de acordo com o modelo de processo proposto pela norma IEEE 12207 4.1.2. Dessa forma este processo é composto por quatro etapas, sendo que cada etapa pode ser subdividida em atividades e caso precise, cada atividade dividida em tarefas. Essas etapas podem ser executadas de forma iterativa, ou sequencial, e são elas:

- Planejamento
- Projeto dos casos de teste
- Execução
- Análise de Resultados

Um esquema geral do processo é apresentado na Figura 5.1 primeiramente da ótica de um macro fluxo no qual cada retângulo representa uma etapa, sendo suas atividades detalhadas posteriormente em micro fluxos nas seções seguintes desse documento.

Esse processo pode ser utilizado para todos os softwares desenvolvidos pela instituição, bem como modificado segundo as necessidades de cada projeto. Cabe ao gerente de teste e/ou projetos verificar essa adequação, e realizar os ajustes pertinentes. Nas seções seguintes são detalhadas cada uma das etapas desse processo.

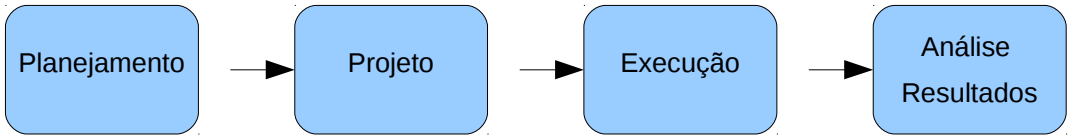

Figura 5.1: *Macro Fluxo do Proceso de Teste.*

5.1 Primeira Etapa: Planejamento

A primeira etapa do processo de teste é conhecida como planejamento de teste. É nesse momento que são abordados pontos sobre o planejamento dos testes e são respondidas questões como: quanto do software e como será testado, quem irá executar os testes, quanto tempo será gasto para cada fase, quais os recursos necessários, qual o tempo total do processo, dentre várias outras questões para tornar o teste possível.

É nessa etapa também em que um ou mais planos de teste são criados, sintetizando as informações necessárias para o desenvolvimento e a execução dos testes de forma adequada. Embora somente um plano de teste possa ser escrito para todo o processo, Rex Black [6] recomenda fortemente que cada fase do processo de teste possua um plano de teste distinto pois cada uma dessas fases possui período de execução, evolução, metodologia e objetivos diferentes.

Os resultados principais da etapa de planejamento são: o plano da atividade de teste, no sentido intangível, e um ou mais planos de testes, que são as representações tangíveis desse plano. No caso da instituição recomenda-se que pelo menos dois documentos de planos de teste para cada produto a ser testado devem ser criados: um para a fase/nível de unidade e um para a fase/nível de sistema.

Fica a cargo do gerente de teste se um terceiro plano de teste será ou não criado. Esse terceiro seria um plano de teste mestre, o qual agruparia os outros dois planos e algumas questões gerais como o cronograma e agenda gerais. Na subseção seguinte são definidos os conteúdos de um plano de teste.

5.1.1 Plano de Teste

O plano de teste, de maneira prática, é um documento o qual contém a representação das ideias para se testar um determinado sistema ou produto de software. Essa representação pode ser pensada em função de três pontos: como os testes são criados, porque eles são criados e como eles serão executados.

No plano de teste podem estar contidos todos os casos de teste - por vezes denominado conjunto de teste - porém [6] recomenda a criação de documentos distintos,

um para o plano de teste em si e outros contendo os casos de teste. Essa é a abordagem escolhida para o processo desse trabalho e é o sugerido para a instituição por questão de organização.

Um bom plano de teste segundo Rex Black [6], é um grande apoio para que o foco do processo em cada fase não seja perdido, e deve tratar detalhadamente dos oito pontos a seguir: escopo, estratégia de teste, equipe, recursos, agenda - cronograma de teste, fases - níveis de teste, orçamento e objetivos principais. A forma que esses conteúdos são apresentados se dá pelo modelo delineado na Tabela 5.1, baseado em [6] e compatível com o padrão IEEE 829 4.1.4. Os detalhes de como preencher cada campo desse modelo são tratados nas subseções seguintes.

Plano de Teste (Modelo)
1. Identificador do documento
2. Introdução
3. Escopo
4. Estratégia
5. Critérios de sucesso/fracasso
6. Critérios de continuação/suspensão
7. Entregáveis
8. Tarefas de teste
9. Requisitos de ambiente
10. Equipe e responsabilidades
11. Agenda/Cronograma
12. Riscos e imprevistos
13. Aprovações
14. Glossário

Tabela 5.1: *Seções Plano de Teste.*

5.1.2 Identificador do documento

Funciona como a identidade do documento podendo ser alfanumérico ou correspondente a padronização da instituição para a nomenclatura de documentos.

5.1.3 Introdução

Apresenta uma visão geral do plano de teste comentando brevemente pontos sobre a execução, metas, objetivos e estratégia utilizada para tal. É necessário privilegiar um linguajar mais descomplicado pois pessoas além da equipe de teste podem entrar em contato com o plano e consequentemente essa introdução deve inteirá-los sobre tal. Essa introdução pode incluir alguns gráficos ou figuras, desde que não seja nada muito longo ou complexo.

5.1.4 Escopo

Define quais a porções do software serão e quais não serão testadas. A definição do que efetivamente deve ser testado pode ser apoiada em relação a três pontos:

- O que pode ser testado, ou seja, todas as áreas que ainda não passaram por testes;
- O que deve ser testado - o que afeta diretamente o usuário, áreas críticas, que limitam ou impossibilitam as pessoas de usarem o software ou de obterem o fim para o qual o produto de software serve como meio;
- E finalmente, o que pode ser testado - das áreas não testadas e críticas, o que o orçamento versus tempo versus equipe disponíveis permitem testar.

Com base na observação desses três pontos, pode-se escolher o que de fato será testado e o que será deixado de fora dos testes, haja vista que o teste de todo o sistema quase sempre é inviável. Para representar essas inclusões/exclusões pode se usar um documento com uma forma tabular.

Outra forma de representar as funcionalidades a serem testadas é listá-las conforme a prioridade. Em todo caso, essa deve ser apenas uma visão geral do que estará e o que não estará incluso nos testes, pois os detalhes desses limites estarão presentes com maior força nos casos de teste.

Opcionalmente, pode-se ainda, empregar uma abordagem de priorização baseada em análise de risco que privilegia o teste daquelas porções do software que, se contiverem defeitos, podem inviabilizar a comercialização ou o sucesso do produto [9].

5.1.5 Estratégia

O item de estratégia apresenta a estratégia/metodologia utilizada nos testes. Existem várias formas de se definir uma estratégia de teste, no entanto uma estratégia que possui destaque no meio de teste pode ser definida baseada nos riscos de cada funcionalidade como sugere Rex Black [6]. Essa estratégia define que para cada grupo de funcionalidades deve-se especificar as atividades principais, critérios e ferramentas que serão utilizadas.

Nesse documento, independente da estratégia utilizada para a priorização das funções e redução de escopo, por questões de início de implantação são propostos os testes unitário e de sistema por meio das técnicas estrutural e funcional respectivamente. Dessa forma, na fase de unidade serão realizados tanto teste caixa branca quanto caixa preta. Já na fase de sistema somente o teste caixa preta será empregado.

Tanto na fase de unidade quanto na fase de sistema, métricas de cobertura de teste devem ser observadas. Dessa forma, ao se fazer o teste unitário utilizando-se a técnica de caixa branca ou preta a cobertura de teste deve ser medida por meio da instrumentação do código. Na fase de sistema a cobertura de código também é medida por meio da instrumentação do código tanto quando da utilização da técnica caixa preta, quando da utilização da técnica exploratória.

A cobertura de código é medida para que seja possível a comparação, em termos de evolução, da variação da cobertura obtida no teste unitário para a cobertura obtida no teste de sistema. Essas informações auxiliam também na identificação de partes específicas do código que foram mais ou menos executadas, e ainda quais partes não puderam ser executadas em nenhuma das fases de teste. Essas medidas em conjunto é o que pode ser entendido por cobertura de teste. Um esquema do relacionamento do produto instrumentado e dos critérios por fase teste está definido na Figura 5.2.

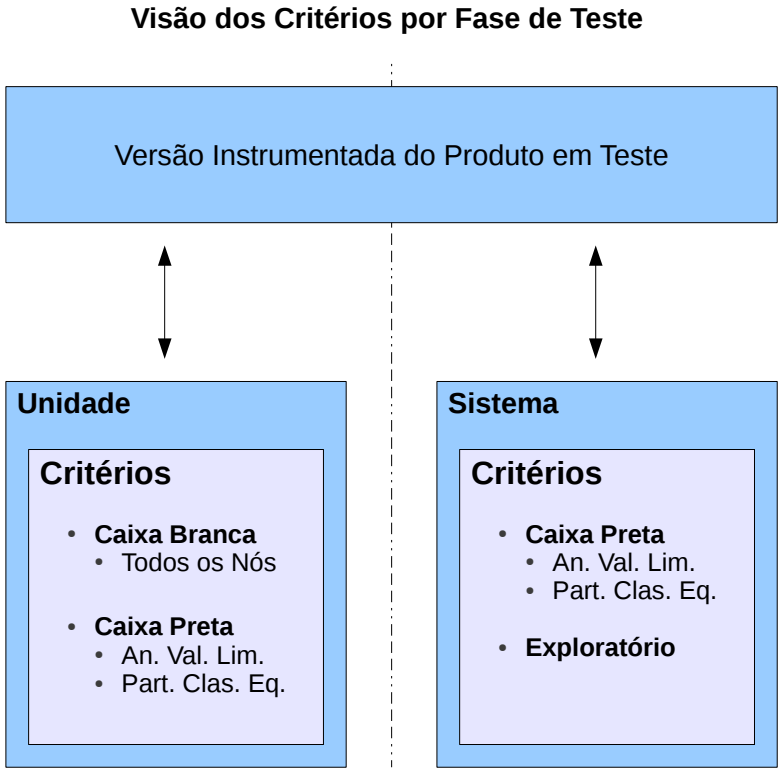

Figura 5.2: *Critérios por Fase de Teste.*

5.1.6 Critérios de sucesso/fracasso

O critério de sucesso define o que deve acontecer para que um sistema possa mudar com sucesso de uma fase de teste para outra, só acontecendo essa mudança caso esse critério seja satisfeito. Caso o sistema falhe nesse critério ele não avança para outra fase.

Em outras palavras, esses são os critérios para saber quando um produto testado passou, ou seja obteve sucesso, ou não numa determinada fase de teste.

5.1.7 Critérios de continuação/suspensão

Define as condições e as situações que devem ocorrer no processo de teste para que esse continue de maneira eficaz e eficiente. Essas condições servem para dizer se uma determinada fase está de acordo com as métricas definidas para tal e se deve ou não continuar os testes até alcançar os objetivos definidos.

Um exemplo disso, é que se for definido que deve ser alcançado 60% de cobertura de uma determinada unidade, os testes devem continuar até que essa porcentagem seja atingida, e tendo atingido essa porcentagem, devem ser suspensos. Ou ainda, se para o teste de sistema, for definido que 50% das funções tidas como essenciais devem ser cobertas, os testes podem ser interrompidos se essa porcentagem de cobertura for atingido.

5.1.8 Entregáveis

Identifica as informações a serem entregues pela atividade de teste, no que se refere a documentos, dados, relatórios de execução, teste, defeitos, etc. Em outras palavras, define os possíveis resultados do plano de teste provindos das execução dos casos de teste. Nesse sentido existem pelo menos três tipos de documentos configurando esses resultados: o primeiro tipo contendo os relatórios de teste, o segundo contendo o relatório de defeitos e o terceiro contendo as métricas de cobertura de código.

5.1.9 Tarefas de teste

Tarefas específicas a serem executadas pelos testadores para que o processo de teste seja possível. Pode tratar de pontos como desenvolvimento e configuração dos ambientes necessários para a execução dos testes, dentre outros.

5.1.10 Requisitos de ambiente

Essa seção do documento é onde se definem os requisitos de hardware, software, rede e espaço necessários para executar os testes. Em outras palavras, em qual espaço

físico o teste será executado, quantas e quais estações deverão estar disponíveis, quais as configurações necessárias para tais, sistemas operacionais, ferramentas, e softwares específicos necessários para a execução e documentação dos testes.

5.1.11 Equipe e Responsabilidades

Define quem e quantos são os integrantes da equipe de teste. Além de quantidades e denominações, nessa seção são definidos os papéis dos participantes. As partes envolvidas nessas responsabilidades podem incluir desenvolvedores, testadores, elementos da equipe operacional ou técnica, administradores de banco de dados e suporte.

5.1.12 Agenda/Cronograma

Define um cronograma - datas e horários - e os objetivos a serem alcançados mediante a execução dos testes da fase em questão.

5.1.13 Riscos e Imprevistos

Também pode ser pensado como "questões em aberto". Define os potenciais eventos que podem dificultar ou tornar impossível a execução do plano de teste. Esse tópico pode abarcar necessidades de treinamento, disponibilidade de recursos extras - humanos, físicos, hardware - dentre outros. Essas preocupações podem ser enxergadas da ótica do gerenciamento de riscos e ocultada do plano de teste caso a instituição possua um gerenciamento de projetos baseado em riscos.

5.1.14 Aprovações

Local reservado no plano de teste para carregar as assinaturas dos interessados (stakeholders) como forma de comprovar a sua aprovação. É uma formalização do documento de forma que garanta que os principais interessados nos testes tenham conhecimento da abordagem que será aplicada.

5.1.15 Glossário

Assim como toda e qualquer área de conhecimento, teste de software é composta por terminologias bem definidas com as quais nem todos os componentes da equipe podem estar confortáveis. Nesse sentido, um glossário deve listar essas terminologias de uma maneira que seja fácil a consulta quando necessário e de forma a garantir que toda a equipe trabalhe a partir de um mesmo conjunto de definições.

5.2 Segunda Etapa: Projeto dos casos de Teste

A ideia de haver separação da etapa de criação dos planos e projeto dos casos de teste é que a atenção de cada etapa seja concentrada em resultados - em nível de documentos - diferentes cada qual com seus detalhes específicos.

É nessa segunda etapa em que os detalhes específicos de como concretizar os testes são definidos. Os objetivo dessa etapa são a definição de um documento que contenha a listagem dos requisitos de teste, e a definição de um conjunto de teste adequado ao critério de teste escolhido. Ser adequado ao critério significa que o conjunto de teste criado deve executar todos os requisitos exigidos pelo critério definido.

Os detalhes de execução são agregados por meio da criação dos casos de teste, sendo eles os principais resultados dessa etapa. De uma maneira geral no contexto da execução dos casos de teste, cada um desses possui três passos principais a serem executados, e são eles:

- **Configurações do caso de teste:** descrição dos passos necessários para configurar o ambiente para executar o caso de teste. Ex: garantir um espaço em disco necessário, conectar um eventual dispositivo de hardware, instalar aplicativos de apoio, etc;
- **Conjunto de condições de teste:** é a execução dos passos do caso de teste em si. O objetivo de se executar um caso de teste é a criação de um conjunto de condições de teste pois são essas condições que permitem ao testador medir a qualidade do sistema em relação a um risco e/ou cenário de uso em particular;
- **Restauração do Estado do Sistema:** são os passos necessários para restaurar o ambiente de teste a condição anterior a configuração realizada para executar esse caso de teste. Ou seja, recolocar o produto em teste em um estado inicial válido para a execução do próximo teste. Ex: liberar o espaço alocado e os aplicativos instalados na etapa de configuração, etc.

Além desses três passos, outras informações como nome mnemônico, identificador do caso de teste, horas consumidas para o teste, dentre outras, são agrupadas em um documento de caso de teste. Essas informações adicionais são necessárias para o gerenciamento do teste de forma a tornar possível a avaliação do processo de teste, bem como uma sistematização do armazenamento dos resultados desses casos de teste.

O modelo a ser seguido para construir um caso de teste é compatível com a norma IEEE 829 4.1.4 e assume que cada passo ou subpasso deve ser provido de: ação a ser realizada, os possíveis dados associados para essa ação, e os resultados esperados. Esse modelo é dividido em três porções de informações sendo elas: cabeçalho, condições de teste e sumário. O esquema geral desse modelo encontra-se delineado na Figura 5.3 e é detalhado nas subseções seguintes.

Modelo Caso de Teste			
Nome do caso de teste:	Nome do caso de teste.		
ID de teste:	Código numérico, número do conjunto de teste, número do teste.		
Conjunto de Testes:	O nome do conjunto de teste ao qual esse caso de teste pertence.		
Prioridade:	Definido a partir da análise de cobertura.		
Requisitos de Hardware:	Hardware necessário para a execução desse caso de teste.		
Requisitos de Software:	Softwares necessários para a execução desse caso de teste.		
Requisito de Teste:	Define o requisito de teste que deu origem ao caso de teste em questão.		
Duração Estimada:	Estimativa de tempo de relógio consumido para essa execução.		
Esforço Estimado:	Estimativa de pessoas por hora de relógio.		
Configurações:	Passos necessários para a configuração do ambiente.		
Restauração do Estado do Sistema:	Passos necessários para retornar o ambiente de teste a condição anterior a execução desse teste.		
ID numérico	**Passo / Subpasso de teste**	**Resultado**	**ID do Defeito**
1	Entrada para passo x; Ação do passo x; Resultado esperado do passo x.		
1.1	Subpasso 1 do passo 1.		
1.2	Subpasso 2 do passo 1.		
2	Nenhum dado é inserido para o passo y; Ação do Passo y; Resultado esperado do passo y.		
Informações de Execução			
Status:	Status geral do caso de teste.		
ID de Configuração de Sistema:	Identificação da configuração de sistema onde esse caso de teste foi executado.		
Testador(es):	Pessoas que executaram esse teste.		
Data de Término:	Data real de término.		
Esforço Efetivo:	Esforço real utilizado para esse caso de teste.		
Duração Efetiva:	Duração real da execução desse caso de teste.		

Figura 5.3: *Modelo Caso de Teste.*

5.2.1 Cabeçalho

A primeira porção do caso de teste compreendida entre os campos "Nome do caso de teste" e "Limpeza de Teste" pode ser vista como o cabeçalho de um texto comum. Nessa parte estão as informações necessárias para identificação do caso de teste, bem como procedimentos de configurações para pré e pós execução, e ainda a definição da equipe que o executará. Os campos que definem essas informações e suas respectivas descrições estão a seguir:

- **Nome do caso de teste:** é um nome em linguagem corrente que descreva a essência desse teste e seja fácil lembrar, por exemplo: teste de stress, teste de desempenho, etc;
- **ID de teste:** identificação numérica e sequencial do caso de teste. Exemplo: 1,2,3...;
- **Conjunto de Teste:** um caso de teste pode pertencer a diferentes conjuntos de teste, embora normalmente ele só pertença a um, esse identificador serve para informar as suas pertinências;
- **Prioridade:** esse campo é opcional, e pode ser definido segundo critério de cobertura ou análise de risco de qualidade. Pode ainda ser definido segundo a experiência do testador e da equipe de teste/desenvolvimento;
- **Requisitos de Hardware:** simples listagem dos aparatos de hardware necessários para a execução do teste;
- **Requisitos de Software:** simples listagem dos softwares necessários para a execução do teste;
- **Requisitos de teste:** critérios de teste geram uma lista de requisitos de teste para serem satisfeitos. Assim sendo, esse campo permite identificar o porquê da criação desse caso de teste, ou seja, a qual requisito de teste ele satisfaz;
- **Duração Estimada:** estimativa de quantas horas de relógio serão necessárias para executar o teste;
- **Esforço Estimado:** estimativa da quantidade de pessoas por hora necessárias;
- **Configurações:** configurações de ambiente necessárias para a execução do caso de teste. Pode ser que nenhuma configuração adicional seja necessária para essa execução, nesse caso esse campo permanece vazio;
- **Restauração do Estado do Sistema:** se alguma modificação foi feita no ambiente no momento da configuração, esses são os passos necessários para desfazer essas modificações, e deixar o ambiente da forma como estava antes. Caso nenhuma configuração tenha sido feita, é necessário ainda procurar por arquivos temporários que possam ter sido gerados durante a execução, do contrário, esse campo também pode ser vazio em alguns casos.

5.2.2 Conjunto de Condições de Teste

A segunda porção de informações diz respeito ao caso de teste em si, ou seja, contém os passos para realizar as condições de teste. Um caso de teste é em essência, uma sequência de passos que definem as ações a ser executadas de maneira sequencial, paralela, ou mesmo em uma forma híbrida dessas duas para que as condições de teste desejadas sejam criadas.

Cada passo pode envolver o uso de dados de teste, inseridos como parte do passo de teste ou no início do caso de teste, e um resultado esperado associado. Um resultado esperado pode ser uma saída, um estado do sistema, ou algum comportamento observável. Há casos em que não existem dados a serem inseridos no sistema em teste, sendo assim, em alguns passos podem estar definidos apenas a ação a ser executada e o resultado esperado.

A organização dos passos se dá de maneira que cada passo ou subpasso possua um identificador numérico. Esse identificador permite que seja feito uma referência específica e inequívoca a qualquer passo ou subpasso desse caso de teste. Essa identificação é especialmente necessária quando da explicação de um defeito em um relatório de defeito ou em qualquer outro contexto.

Na coluna de resultados dos passos o testador irá anotar os resultados de cada passo executado nesse caso de teste, de forma que os possíveis resultados, segundo Rex Black [6], são os seguintes:

- **Sucesso:** o testador executa o passo ou subpasso e obtém o resultado esperado e nada além dele. Isso quer dizer que o passo/subpasso não encontrou nenhum defeito e terminou de forma correta;
- **Atenção:** o testador executa o passo ou subpasso e obtém em maiores ou menores proporções um resultado inesperado. Esse rótulo é definido somente caso não tenha acontecido algum estado de erro fatal como por exemplo: travamento total da cpu, reinícios, ou inviabilidade total ou parcial de uso do ambiente, caso contrário é rotulado como **Falha**. O rótulo **Atenção** não é o mesmo que **Falha**, pois apesar do resultado não ter sido o esperado, em atenção o caso de teste cumpriu seu papel e foi útil para identificar algum defeito.
- **Falha:** o testador executa o passo ou subpasso e obtém um resultado drasticamente inesperado. Diferentemente do caso de **Atenção**, nesse momento o testador observou grandes problemas desencadeados devido a essa execução. Dessa forma, ele o marca como **Falha**.
- **Bloqueado:** o testador não executou o passo ou subpasso pois foi impedido por algum defeito conhecido, falta de recursos, ou qualquer outro fator bloqueante que deverá ser colocado no caso de teste como explicação para tal bloqueio;

- **Não Executado:** o testador não executou o passo ou subpasso. Essa marcação é definida quando o testador escolher não executá-los por algum motivo que deve ser plausível e explicado no caso de teste.

Ao se marcar um passo/subpasso de teste como **Falha** ou **Atenção** o testador deve anotar na coluna de ID do Defeito - Identificação do defeito - um identificador numérico para tal defeito encontrado. Esse identificador servirá para registrar e encontrar o relatório desse defeito no sistema de rastreamento de defeitos.

Informações adicionais sobre os defeitos tais como classificação, isolamento, dentre outras, são detalhadas no relatório de defeito para cada um dos defeitos encontrados. Os componentes do relatório de defeito e como fazê-lo estão detalhados na seção 5.3.1.

5.2.3 Sumário

A terceira porção de informações do caso de teste é um resumo das informações de execução desse caso de teste e possui seis campos para tal. O primeiro desses campos é o status, e uma vez que um caso de teste é uma composição de passos de teste, esse status está intimamente ligado aos status individuais desses passos. Dessa maneira, o status geral do caso de teste pode ser rotulado com um, dentre os seguintes possíveis:

- **Em andamento:** se algum passo ou subpasso dele ainda não foi completado;
- **Bloqueado:** se algum passo ou subpasso está bloqueado então o status do caso de teste é bloqueado;
- **Falha:** se algum passo ou subpasso falhou o status do caso de teste recebe o status de falha;
- **Sucesso:** se todos os passos obtiveram sucesso em suas execuções.

Importante notar que não se pode classificar o status do caso de teste como **Não Executado**, uma vez que essa é uma decisão em nível de passo ou subpasso de teste, não sendo aplicável para um caso como um todo. Além do status geral, nessa terceira parte do caso de teste devem ser registrados as seguintes informações adicionais em outros cinco campos:

- **ID de Configuração de Sistema:** anotações sobre configurações específicas de sistema utilizadas para esse teste;
- **Testador(es):** nomes de todos os testadores envolvidos;
- **Data de Término:** data de quando o teste foi terminado;
- **Esforço Efetivo:** esforço efetivamente gasto em pessoas por hora utilizadas, ou seja, a soma de todos os esforços dos passos/subpassos que foram de fato utilizados para esse caso de teste;

- **Duração Efetiva:** duração efetiva em horas, isso é, a soma de todas as durações de todos os passos/subpassos que foram de fato utilizadas para esse caso de teste.

Essas informações adicionais servem principalmente para permitir a reprodutibilidade dos casos de teste. Alem disso, as componentes sobre sobre datas, tempo e esforços são importantes na hora de avaliar o quanto da atividade de teste foi cumprida ao passo que permitem acompanhar de maneira numérica a evolução das execuções de cada caso de teste.

5.2.4 Cobertura de Teste

Existe a necessidade de uma especificação de cobertura ou outro método para garantir a suficiência do teste, ou seja, para dizer quando os testes devem parar. Nesse contexto, e de maneira simplificada, cobertura de teste pode ser definida como "até quanto um software será testado". Em outras palavras, é dizer o quão abrangente é a atividade de teste e quais as funcionalidades de fato são ou não abarcadas pela atividade de teste.

O tipo de cobertura que é relevante varia de acordo com a fase - unidade, sistema - de teste. Dessa forma no teste de unidade a cobertura geralmente é expressa em termos de porcentagem de código testado enquanto no teste de sistema a cobertura pode ser a porcentagem dos requisitos testados.

A elaboração da cobertura de teste não é estática podendo mudar de acordo com a execução dos testes. Essa mudança se dá pois as necessidades dos clientes podem mudar de acordo com o próprio processo de desenvolvimento. Esse dinamismo é característico até mesmo da elicitação dos requisitos, o que ocasiona mudanças também no foco dos testes.

Para não haver equívocos na escolha da cobertura as considerações de parada devem ser estabelecidas com base nas maiores necessidades do cliente. Isso quer dizer, que as funcionalidades mais importantes para ele devem ter maior prioridade no momento da confecção dos casos de teste. Essa priorização garante que as funcionalidades que estão sendo testadas são as que o cliente irá sentir maior diferença, influenciando positivamente na qualidade percebida por ele.

Existem variadas estratégias para se analisar cobertura, dentre elas pode-se destacar: baseada em números, definindo-se a porcentagem aceitável de cada parte a ser testada, por análise de risco de qualidade, ou utilizando-se uma abordagem denominada rastreamento de funcionalidades ou matriz de função-teste.

No processo proposto as métricas baseadas em números e rastreamento de funcionalidades são utilizadas. Sendo assim, a cobertura baseada em números é definida para a fase de unidade. Já para a fase de sistema, tanto a métrica baseada em números

quanto o rastreamento de funcionalidade são previstas. Os detalhes dessa cobertura estão descritos na seção 5.1.5.

Quando o documento de requisitos e as especificações da arquitetura do software estiverem disponíveis eles podem ser utilizados de maneira a facilitar a definição da cobertura de teste. Isso é feito de forma que a partir deles seja criado o documento de referências cruzadas, no qual para cada funcionalidade descrita no documento de requisitos seja identificado um ou mais casos de teste que sejam responsáveis por testar essa funcionalidade.

Quando o documento de requisitos e as especificações da arquitetura não estiverem disponíveis - para produtos legados, feitos mediante algumas metodologias ágeis ou mesmo desenvolvidos sem um deles por exemplo - uma abordagem semelhante a com a existência do documento de requisitos pode ser utilizada. Essa abordagem consiste em se listar as funcionalidade gerais do produto a ser testada, como se fosse a construção do documento de requisitos aplicando uma engenharia reversa no produto. De posse dessa listagem das funcionalidades, tenta-se parear funcionalidade com caso de teste da mesma maneira que na forma anterior.

Para as funcionalidades que não possuam casos de teste correspondentes deve-se avaliar a necessidade de construção de casos de teste para elas ou se elas podem ser deixadas de fora do teste. A cobertura então - em ambos os casos, com ou sem documento de requisitos - é definida por meio de iterações desses passos até que uma determinada porcentagem do produto seja atingida ou até que pelo menos as funcionalidades de maior importância sejam cobertas.

Com base nessas observações pode-se concluir que cobertura de teste é uma propriedade emergente dos conjuntos de teste, e não somente a um caso de teste em si, podendo ser avaliada por meio das várias fases de teste.

5.3 Terceira Etapa: Execução dos Testes

A etapa de execução dos testes é o momento em que as ações são executadas sobre o programa em teste. Essas ações são aquelas detalhadas pelos passos dos casos de teste. Nesse contexto, é nessa etapa em que os casos de teste são colocados em prática permitindo a observação das reações e comportamentos do software em relação a esses testes.

Os resultados das execuções dos casos de teste devem ser documentados para que seja possível a reprodução desses resultados. Dessa forma, sintetizando as informações observadas e os defeitos encontrados com essa execução, os relatórios de defeito configuram os principais produtos resultados dessa etapa. Adicionado ao relatórios de defeitos,

outras documentações podem ser delineadas dependendo da necessidade, como por exemplo relatórios de execução (logs), e também os relatórios de teste.

Para a execução dos testes, duas questões principais devem ser levantadas para que se tenha uma boa estimativa do tempo que levará para serem executados. A primeira é quanto tempo leva para executar a massa de teste completa - denominado tempo para cada passo. A segunda questão é quantas vezes é necessário executar essa massa de teste sobre as versões que serão disponibilizadas para que se encontrem os defeitos importantes - o que é denominado número de ciclos. Essas duas questões devem ser respondidas pela etapa de planejamento e influenciarão diretamente no andamento da execução dos testes.

O sistema de teste é formado por conjuntos de teste, os quais, por sua vez, são formados por casos de teste cada qual responsável por partes da cobertura do código. Ao se criar os casos de teste, e os conjuntos de teste, prioridades de execução são definidas para ambos. Com base nessas prioridades, uma estratégia para definir quais testes serão executados e quando, é organizar uma execução que privilegie essas prioridades. Desse modo, os casos de teste mais importantes, ou seja, os responsáveis pelas funcionalidades que mais importam aos clientes, serão executados primeiramente e por mais vezes que os outros casos de teste.

Nesse trabalho são propostos os testes unitário e de sistema por meio das técnicas estrutural e funcional, respectivamente. Ao se fazer o teste unitário utilizando-se a técnica de caixa branca devem ser observadas as métricas para a cobertura do código definidas no plano de teste. Essa cobertura é medida por meio da instrumentação do código. Na fase de sistema, utiliza-se a técnica caixa preta, e a cobertura de código também deve ser medida, com base nessa mesma abordagem de instrumentação.

Uma importante observação deve ser feita ao se executar os testes, que é a seguinte: a execução em cada fase do teste - unidade ou sistema - deve começar e terminar passando por toda a massa de teste. Ou seja, essa observação enuncia a necessidade de executar todos os casos de teste de todos os conjuntos de teste criados para um determinado nível, tanto no passo inicial quanto final da etapa para esse nível.

Embora em alguns passos de uma etapa sejam executados apenas alguns conjuntos de testes, como por exemplo os com maior prioridade, é grande a necessidade de uma etapa iniciar e terminar passando por toda a massa de teste. A justificativa dessa necessidade é apoiada na premissa de que, com a primeira execução de toda essa massa seja possível que a equipe de teste tenha uma visão geral dos defeitos que serão apresentados pelo produto em teste. Adicionando-se a isso, a reexecução ao final deve tornar possível uma visão de como os defeitos evoluíram, e foram corrigidos, possibilitando assim se necessário, impedir a liberação de uma versão com muitos defeitos para o usuário.

A execução dos testes ocorre em em ciclos, isso quer dizer que a cada versão liberada do produto, a massa de teste deve ser novamente executada sobre ele e as

anotações sobre suas saídas e outras informações de execução devem ser realizadas. A partir da execução dos testes para cada defeito encontrado um relatório de defeito deve ser criado e inserido no sistema de rastreamento de defeitos.

Cada caso de teste, como enunciado na seção 5.2 agrupa os passos para a criação das condições de teste bem como as informações de apoio para essas condições. Ao se executar os casos de teste, uma determinada sequencia lógica para essa execução deve ser obedecida para que se obtenha os resultados desejados. Dessa forma, um esquema geral para essa execução pode ser encontrado na Figura 5.4.

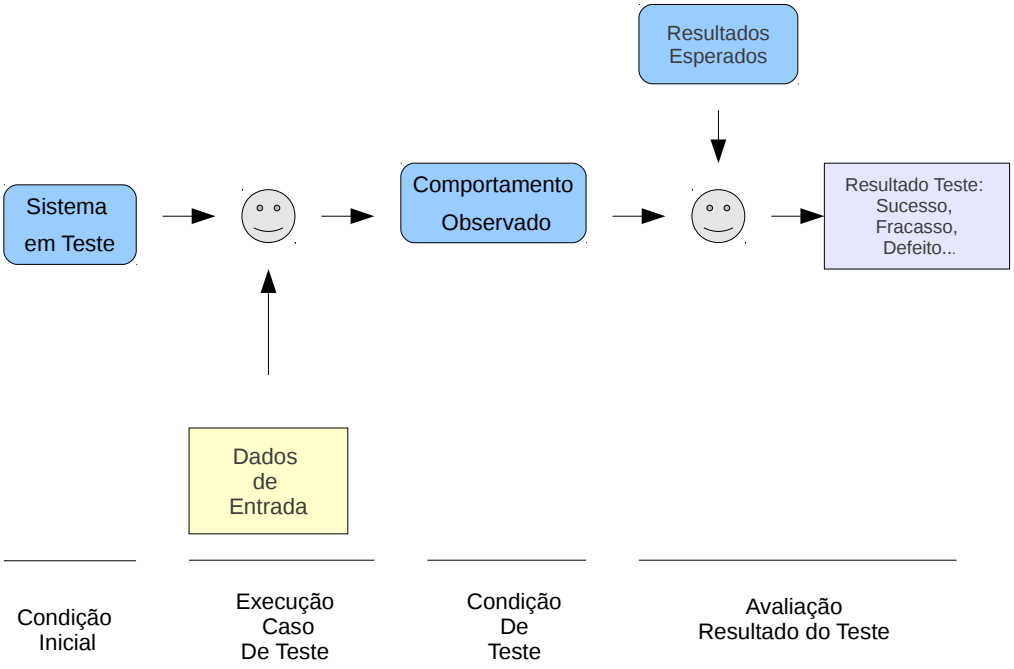

Figura 5.4: *Esquema Genérico Execução de Caso de Teste.*

5.3.1 Relatório de Defeito

O relatório de defeito é um documento técnico que descreve vários comportamentos diferentes do esperado ou tipos de falha associados a um determinado defeito. Dessa forma, para cada defeito encontrado por um caso de teste esse deve ser registrado num sistema de armazenamento de defeitos nomeado por um identificador único e inequívoco juntamente com informações relevantes sobre ele, configurando o relatório de defeito.

O relatório de defeito é o produto mais tangível da atividade de teste, sendo ainda o produto mais importante dessa atividade. Essa importância que lhe é conferida provem de suas funções, que são a de registrar as origens dos defeitos e promover a comunicação desses defeitos entre as equipes de teste e desenvolvimento.

Um bom relatório de defeito auxilia a gerência de projetos proporcionando as informações necessárias para essa decidir quando e como retificar os problemas encontrados. Esse relatório também deve possuir informações detalhadas o suficiente para que um desenvolvedor consiga reproduzir e corrigir com sucesso o defeito encontrado. Dessa forma as informações contidas nos relatórios de defeito funcionam como um guia para a equipe de desenvolvimento a qual realizará as correções necessárias no produto.

É importante que os relatórios de defeitos sejam escritos paralelamente a execução dos testes. Isso ocorre, pois se deixados para depois, valorosos detalhes sobre os defeitos podem ser esquecidos, ou mesmo descritos de maneira incompleta, comprometendo a qualidade desses relatórios e consequentemente, a qualidade do produto testado.

Estruturalmente falando, o relatório de defeito é dividido em três porções de informações básicas sendo elas: cabeçalho, passos para reprodução do defeito e isolamento. Um modelo adaptado de [6] está definido pela Figura 5.5 e seus campos são comentados nas subseções seguintes.

Modelo de Relatório de Defeito						
ID do Defeito	**Nome do Projeto**	**Testador**	**Data de Criação**	**Prioridade**	**Classe**	**Estado**
Identificação do defeito.	Nome do projeto que esse defeito pertence.	Nome de quem encontrou o defeito.	Data em que esse relatório foi criado.	Prioridade com que esse defeito deve ser reparado.	Classe do defeito.	Estado do defeito.
Sumário						
Descrição concisa da essência do defeito.						
Passos para Reprodução do Defeito						
Passos necessários para que defeito seja reproduzido. Exemplo: 1. Primeiro passo; 2. Segundo passo; 3. Terceiro passo; n . Enésimo passo.						
Isolamento						
Resultados, informações e limites do defeito.						

Figura 5.5: *Modelo Relatório de Defeitos.*

5.3.1.1 Cabeçalho

A primeira porção de informações do relatório de defeito é o cabeçalho. Essa parte é composta por oito itens os quais possuem a função de identificar e resumir as informações gerais sobre o relatório de defeito. A seguir são comentados brevemente cada um desses itens:

- **ID do Defeito:** campo numérico, ou alfanumérico com a mesma identificação presente no caso de teste em que foi encontrado. Essa é a maneira com que os casos de teste estão ligados aos relatórios de defeito;
- **Nome do Projeto:** define o nome do projeto ao qual esse relatório de defeito pertence;
- **Testador:** nome da pessoa que criou esse relatório de defeito;
- **Data de Criação:** data em que o relatório foi criado;
- **Prioridade:** esse campo define a prioridade do relatório de defeito, e consequentemente a prioridade do defeito em si. Essa prioridade auxilia na definição de uma ordem para correção dos defeitos.;
- **Classe:** esse campo define a classe a qual o defeito pertence. Sendo assim, tem como função o agrupamento dos defeitos em determinadas classes, como por exemplo, defeito de interface, ou defeito em lógica, etc;
- **Estado:** define o estado desse relatório e consequentemente do defeito. Sendo que esse estado pode ser atribuído, suspenso ou corrigido;
- **Sumário:** descrição simples, completa e direta da essência do defeito de modo que ao ler essa descrição, o leitor possua uma visão do que se trata esse relatório.

5.3.1.2 Passos para Reprodução

A segunda porção de informações configura a descrição dos passos necessários para que o defeito seja reproduzido. Essa é a seção mais importante do relatório pois ela será utilizada pelos desenvolvedores para reproduzir o defeito, e consequentemente, repará-lo. Portanto, esse item deve ser escrito com maior esmero para que o defeito não seja desacreditado pelo programador caso ele não consiga reproduzir o defeito por alguma falha de explicação desses passos.

5.3.1.3 Isolamento

A terceira porção do relatório de defeito agrupa as informações observadas pelo testador que confirmam a existência desse defeito. Sendo assim, nesse item são descritos os fatores que afetam a manifestação desse defeito, ou seja, os limites desse defeito. Exemplos de informações que podem estar descritas nesse item são: algum programa

que influencie na ocorrência desse defeito, ou ainda, se esse defeito ocorre somente em um sistema operacional específico, ou uma versão de navegador, dentre outros.

5.3.2 Relatório de Teste

Os relatórios de defeitos possuem a função de agrupar os defeitos encontrados com as informações sobre eles, de modo que esses defeitos sejam registrados no sistema de rastreamento de defeitos para fins de gerenciamento. No entanto, esses documentos dizem respeito somente aos defeitos não abarcando as informações sobre os casos de teste que foram corretos, ou outros pontos sobre a execução dos casos de teste essenciais para se ter uma medida de qualidade do produto. É com base na necessidade dessas informações que o relatório de teste é motivado a existir.

O relatório de teste sintetiza todas as informações de execução dos testes e pode ser desenvolvido com foco em uma fase, configurando o relatório de teste de uma determinada fase, ou com foco em toda a atividade de teste, denominado o relatório mestre de teste. De acordo com o processo proposto nesse documento, recomenda-se a criação de pelo menos dois relatórios de teste distintos: um para a fase de unidade, e um para a fase de sistema. A criação do terceiro relatório, o relatório mestre, fica a cargo do gerente de teste da instituição.

5.4 Quarta Etapa: Análise dos Resultados

A etapa de análise dos resultados é o momento em que são analisados os resultados de execução dos testes, sendo que essa análise é feita ao final de cada fase - unidade, ou sistema - do processo de teste e também ao final do processo completo.

As informações necessárias para essa avaliação são os documentos gerados ao longo de cada fase, consistindo dos relatórios de execução, defeitos, teste, dentre outras possíveis resultados das execuções dos testes.

Além de manter os o registro dos defeitos, e seus respectivos relatórios organizados, o sistema de rastreamento de defeitos pode proporcionar vários tipos de métricas com base nas informações contidas nesses relatórios. Essas informações podem, e devem ser utilizadas na avaliação dos resultados do processo como um todo auxiliando a tomada de decisões quanto a mudanças no processo.

No processo proposto, as métricas a serem avaliadas nessa etapa são a cobertura de teste, ou seja, quais das funcionalidade foram de fato cobertas pelos casos testes executados e também a cobertura de código, ou seja, quanto do código foi efetivamente coberto em cada fase do processo de teste.

São essas informações que fornecerão uma maneira de avaliar a qualidade dos testes realizados, e que refletirão na qualidade do produto em desenvolvimento. Essas medidas são portanto uma ferramenta para o gerente de teste. Elas o auxiliam a decidir quais as modificações devem ser realizadas no processo de teste ou em cada fase específica caso algumas das métricas auferidas não sejam satisfatórias, ou caso algum outro problema seja identificado.

CAPÍTULO 6

Conclusão

A qualidade dos produtos de software produzidos por uma instituição está intimamente ligada a qualidade de seu processo de desenvolvimento de software. Em face disso, e em busca da melhoria dos produtos produzidos, é que o teste de software se torna uma importante ferramenta na medição e na melhoria da qualidade nas instituições desenvolvedoras.

Como detalhado por todo esse documento, a atividade de teste de software possui como objetivo principal encontrar defeitos, ou seja, fazer o produto falhar. Mas não é somente fazê-lo falhar a esmo, os passos, dados, bem como toda informação que levaram a essa falha devem estar meticulosamente documentados, para que sejam reproduzíveis. E ainda, os defeitos encontrados devem ser documentados com idem esmero, e registrados em um sistema de rastreamento de defeitos para que seja possível aos gerentes de teste e projetos obter métricas a partir deles.

Além dessa identificação de defeitos, a atividade de teste tem como objetivo garantir que o produto desenvolvido é o correto - ao verificar se está de acordo com os requisitos - e ainda se o produto foi desenvolvido corretamente - se a quantidade de código coberta e em funcionamento é o razoável de acordo com os critérios da instituição.

Ainda sobre o processo de teste, o correto é que esse processo seja desenvolvido em paralelo com o processo de desenvolvimento. Esse paralelismo pode então permitir auferir a qualidade tanto do produto, quanto do processo em momento de desenvolvimento, de maneira que seja possível realizar alterações necessárias a ambos em tempo hábil, e com o menor prejuízo possível.

Propor um processo de teste é sem dúvida uma tarefa complexa, na qual cada detalhe é de suma importância para o resultado esperado. Nesse sentido, é necessário que sejam definidos desde os papéis de uma equipe, a infraestrutura necessária para tal, até os documentos que serão gerados no momento de teste, em detalhes.

Com todo esse panorama exposto, se espera que a base de conhecimento definida, juntamente com as explicações de cada componente de tal, e ainda, o processo de teste de software proposto, possam contribuir positivamente de maneira efetiva e efi-

caz no que se refere ao aumento da qualidade dos produtos de software produzidos pela instituição.

6.1 Trabalhos Futuros

A base de conhecimento definida nesse documento, juntamente com o processo de teste proposto formam os alicerces para a implantação da atividade de teste no CERCOMP. Tendo esse material como ponto de partida, os próximos passos para se chegar ao sucesso dessa implantação são:

- identificação dos níveis de integridade da instituição;
- levantamento de quantos, quais são, e em quais linguagens os produtos de softwares da instituição são desenvolvidos, haja vista que o último levantamento data do primeiro semestre de 2011;
- definição de ferramentas que sejam compatíveis com testes de unidade e sistema para as linguagens utilizadas;
- sincronização do processo de teste com o processo de desenvolvimento da instituição;
- definição de uma equipe de teste;
- definição de infraestrutura necessária para os testes;
- definição dos papéis dos membros da equipe de teste;
- treinamento dos membros da equipe de teste;
- implantação do processo em produtos de software pilotos bem definidos;
- acompanhamento da execução dos pilotos;
- avaliação dos resultados da aplicação desse processo nesses produtos;
- sugestão de melhorias nos processos de desenvolvimento e teste de software baseado nas métricas obtidas por meio dos pilotos;
- e por fim, expansão da implantação do processo de teste proposto.

É importante observar que esses trabalhos futuros não são somente etapas sequenciais e bem definidas como pode passar a impressão pela listagem algorítmica supracitada. Isso ocorre devido a própria natureza da melhoria dos processos, que é iterativa. Sendo assim, se torna necessário enxergar essa exposição de passos como foco no objetivo principal, que é a implantação em maior amplitude possível do processo de teste. Portanto, esse é um processo contínuo e não pouco complexo que demanda tempo e esforços não triviais, mas que com a devida atenção e dedicação se torna perfeitamente possível.

Referências Bibliográficas

[1] **Iso 9126 - 1 - software quality characteristics**, 2003.

[2] **Ieee 1012 - standard for software verification and validation**, 2004.

[3] **Ieee 829 standard for software and system test documentation**, 2008.

[4] **Iso/iec 12207 - systems and software engineering - softwarelife cycle processes**, 2008.

[5] AMMANN, P.; OFFUTT, J. **Introduction To Software Testing.** Cambridge University Press, New York, USA, 2008.

[6] BLACK, R. **Managing The Testing Process.** Wiley Publishing, New York, USA, 2002.

[7] CRESPO, A.;AND MARTINEZ, M. J. M.; JUNIOR, M. **Application of the IEEE 829 Standard as a Basis for Structuring the Testing Process**. *Journal of Software Testing Professionals*, (12):13–17, December 2002.

[8] DELAMARO, M. E.; MALDONADO, J. C. J. M. **Introdução ao Teste de Software.** Campus Elsevier, Rio de Janeiro, RJ, Brasil, 2007.

[9] GERRARD, P.; THOMPSON, N. **Risk Based E-Business Testing**. Artech House, Inc., Norwood, MA, USA, 2002.

[10] J MYERS, G. **The Art of Software Testing.** Wiley & Sons Publishing, New York, USA, 2004.

[11] PRESSMAN, R. **Engenharia de Software.** McGraw-Hill, São Paulo, SP, Brasil, 2006.

[12] SOMMERVILLE, I. **Software Engineering.** Addison Wesley, England, 2007.

[13] TIAN, J. **Software Quality Engineering.** John Wiley and Sons., New Jersey, USA, 2005.

APÊNDICE A

Relações de Entradas e Saídas para cada Tarefa de Teste em cada Fase de Desenvolvimento de Acordo com IEE 829

1. Análise de Requisitos		
Tarefas de Teste	**Entradas**	**Saídas**
1. Gerar Plano de Teste de Sistema	Requisitos de Sistema Matriz de Rastreabilidade de Teste	Plano de Teste de Sistema
2. Criar Matriz de Rastreabilidade de Teste	Requisitos de Sistema	Matriz de Rastreabilidade de Teste

2. Design		
Tarefas de Teste	**Entradas**	**Saídas**
1. Gerar Design de Teste de Sistema	Plano de Teste de Sistema	Design de Teste de Sistema
2. Gerar Plano de Teste de Unidade	Requisitos de Unidade	Plano de Teste de Unidade
3. Gerar Design de Teste de Unidade	Plano de Teste de Unidade	Design de Teste de Unidade
4. Atualizar Matriz de Rastreabilidade de Teste	Matriz de Rastreabilidade de Teste	Matriz de Rastreabilidade de Teste Atualizada

3. Implementação		
Tarefas de Teste	**Entradas**	**Saídas**
1. Gerar Casos de Teste de Sistema	Requisitos de Sistema Plano de Teste de Sistema Design do Teste de Sistema	Casos de Teste de Sistema
2. Gerar Casos de Teste de Unidade	Requisitos de Software Plano de Teste de Unidade Design do Teste de Unidade	Casos de Teste de Unidade
3. Executar Teste de Unidade	Plano de Teste de Unidade Design de Teste de Unidade Casos de Teste de Unidade Dados de Teste de Unidade	Resultados Teste de Unidade Relatórios de Defeitos Entradas para Relatório do Teste de Unidade
4. Avaliar dos Resultados do Teste de Unidade	Resultados dos Testes de Unidade Relatórios de Defeitos	Relatórios de Defeitos Análise dos Resultados dos Testes de Unidade
5. Preparar o Relatório de Teste de Unidade	Análise dos Resultados dos Testes de Unidade Relatórios de Defeitos	Relatório de Teste de Unidade
6. Atualizar Matriz de Rastreabilidade de Teste	Matriz de Rastreabilidade de Teste	Matriz de Rastreabilidade de Teste Atualizada

4. Teste		
Tarefas de Teste	**Entradas**	**Saídas**
1. Executar Teste de Sistema	Plano de Teste de Sistema Design de Teste de Sistema Casos de Teste de Sistema Dados de Teste de Sistema	Relatórios de Defeitos Resultados dos Testes de Sistema
2. Avaliar resultados dos Testes de Sistema	Plano de Teste de Sistema Resultados dos Testes de Sistema Relatórios de Defeitos	Relatórios de Defeitos Análise dos Resultados dos Testes de Sistema
3. Preparar Relatório de Teste de Sistema	Análise dos Resultados dos Testes de Sistema Relatórios de Defeitos	Relatório de Teste de Sistema