VANESSA MAIRA DE PAIVA SILVA

SWTEST: UM PROCESSO DE TESTE DEFINIDO PARA UMA EMPRESA DE PEQUENO PORTE DESENVOLVEDORA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2006

VANESSA MAIRA DE PAIVA SILVA

SWTEST: UM PROCESSO DE TESTE DEFINIDO PARA UMA EMPRESA DE PEQUENO PORTE DESENVOLVEDORA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Engenharia de Software

Orientador:

Prof. Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS – BRASIL
2006

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Silva, Vanessa Maira de Paiva

SwTest: Um Processo de Teste Definido para uma Empresa de Pequeno Porte Desenvolvedora de Software/ Vanessa Maira de Paiva Silva – Minas Gerais, 2006.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Engenharia de Software. 2.Processo de Teste de Software. 3. Qualidade de Software. 1. SILVA, V. M. P.. II. Universidade Federal de Lavras. III. Título.

VANESSA MAIRA DE PAIVA SILVA

SWTEST: UM PROCESSO DE TESTE DEFINIDO PARA UMA EMPRESA DE PEQUENO PORTE DESENVOLVEDORA DE SOFTWARE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 22 de Setembro de 2006.

_____
Prof. André Luiz Zambalde

_____
Prof. Marluce Rodrigues Pereira

_____
Prof. Heitor Augustus Xavier Costa
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2006

Dedico este trabalho aos meus pais.

Sem vocês nada disso teria razão. Amo vocês.

AGRADECIMENTOS

Agradeço,

A Deus.

As duas pessoas mais importantes da minha vida, Maria e Hermínio, meus pais, pelo amor, confiança, dedicação e incentivo. Amo vocês. Serei eternamente grata. Obrigada pelos vários ensinamentos e bons exemplos.

Aos meus irmãos Karina, Gui (Guilherme) e Bruna, vocês são maravilhosos!

A minha sobrinha e afilhada amada, Larissa.

Aos parentes que sempre tiveram uma palavra de carinho e incentivo, vocês fazem parte dessa conquista, em especial a vó Lurdinha, ao vô Flávio (*in memorian*), a Neuza, ao Célio, a Rita, a Nininha e ao Jaime, que me ensinaram que o significado de família transcende os laços de sangue. Amo vocês.

Aos meus amigos, em especial:

A Bia e ao Sandro, pela cumplicidade e pelo companheirismo nos diversos acontecimentos em nossas vidas;

As companheiras e amigas de república Cibele, Juliana, Camila, Paulinha, Carmem Lúcia e Nanda;

As meninas da república Whikas, Gicely, Larissa e Natália, por todas as risadas e farras, vocês são inesquecíveis;

Aos amigos queridos Breno, Filipe e Leandro e demais colegas de sala.

Sem vocês não teria graça!

Ao Gu (Gustavo Lopes Fogaça de Mello) que dedicou seu amor, sua amizade e sua compreensão em todos os momentos que tivemos juntos. Você é meu anjo, meu amigo e minha paixão. Eu te Amo! Beijo na Boca!

A Ana Cristina Rouiller e a família SWFactory, que me deram oportunidade e acreditaram em meu trabalho, obrigada pela convivência e aprendizagem, vocês são maravilhosos.

Por último, agradeço ao meu orientador, o professor Heitor Augustus Xavier Costa, que me ajudou a concretizar esse trabalho e aos demais professores do Departamento de Ciência da Computação da UFLA.

SWTEST: UM PROCESSO DE TESTE DEFINIDO PARA UMA EMPRESA DE PEQUENO PORTE DESENVOLVEDORA DE SOFTWARE

RESUMO

Este trabalho apresenta o desenvolvimento do SwTest, um processo de teste de software definido para uma empresa de pequeno porte desenvolvedora de software. O SwTest se baseia nas normas IEEE 829 e ISO/IEC 12119. A importância em produzir software com qualidade e a necessidade de profissionais na área de testes de software direcionou a elaboração de um processo que contemple as etapas de planejamento, projeto, execução e avaliação das atividades de teste. O entendimento dos conceitos de engenharia, qualidade de software e teste de software tornou-se essencial para a concepção do trabalho. O processo SwTest se mostrou uma alternativa viável para melhorar e garantir a qualidade do software desenvolvido de forma gradual e voltado à realidade da empresa de pequeno porte desenvolvedora de software.

Palavras-chave: processo de teste de software, qualidade de software, estratégias de testes, testes.

SWTEST: A PROCESS OF TEST DEFINED FOR A SMALL BUSINESS COMPANY OF SOFTWARE

ABSTRACT

This work presents the development of the SwTest, a process of test of software defined for a small business company of software based on the IEEE 829 norms and ISO/IEC 12119. The importance in producing software with quality and the necessity of professionals in the area of software tests directed the elaboration of a process that contemplates the stages of planning, project, execution and evaluation of the activities of test. The agreement of the concepts of engineering, quality of software and test of software became essential for the conception of the work. The SwTest process if showed as a viable alternative to improve and to guarantee the quality of the developed software of form gradual and directed to the reality of the small business company of software.

Key-Words: process of software test, quality of software, strategies of tests, tests.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A realização de testes de software envolve atividades de planejamento, execução dos testes e avaliação dos resultados obtidos. No entanto, para que os testes sejam eficientes, os artefatos gerados durante o desenvolvimento devem conter informações relevantes para a realização das atividades relacionadas com o teste. Desta forma, é possível preparar e executar testes sistematicamente contribuindo para que os erros sejam identificados logo no início do processo de desenvolvimento e diminuindo os gastos necessários para a sua correção.

O presente trabalho apresenta a importância da realização dos testes de software durante todo o ciclo de vida do software em construção.

1.1 Motivação

O desenvolvimento de software envolve uma série de atividades de produção em que as oportunidades para introduzir falhas humana são grandes. Erros podem ocorrer no início do processo, onde os objetivos podem ser errônea ou imperfeitamente especificados, bem como em estágios posteriores de projeto e desenvolvimento.

Em função da inabilidade humana de realizar e de se comunicar com perfeição, o desenvolvimento deve ser acompanhado por uma atividade de garantia de qualidade.

A eficiência e a qualidade têm sido almejadas em todas as áreas de produção e serviços e o software, como um produto comercial, deve ser passível dos referidos adjetivos.

No cenário brasileiro de desenvolvimento de software, observa-se nos últimos anos, uma crescente conscientização por parte das empresas com relação à qualidade de seus produtos. No entanto, a qualidade de um software não pode ser imposta após a sua finalização, estando ela fortemente relacionada à qualidade do processo de software.

Uma das maneiras de controlar a qualidade, tanto do produto como do processo, é por meio do teste de software, que define um conjunto de atividades com o objetivo de encontrar problemas no software e no processo (Sommerville, 2003).

A palavra Teste é derivada do latim (*testum*), significando originalmente um pote de barro onde metais eram decompostos para determinação da presença de diversos elementos ou medição de seu peso[1]. Daí surgiu à expressão “colocar em teste”. Hoje, o uso da palavra difundiu-se. Mas, a situação se complica na hora de ampliar o conceito, passando-o de uma série de perguntas ou exercícios voltados para a medição do conhecimento ou das habilidades das pessoas ao processo de avaliação de software**.**

No dicionário Aurélio (Ferreira, 1986), encontra-se para o verbete teste: *"S. m. 1. Exame, verificação ou prova para determinar a qualidade, a natureza ou o comportamento de alguma coisa, ou de um sistema sob certas condições. 2. Método, processo, procedimento ou meios utilizados para tal exame, verificação ou prova."*

Segundo (Pressman, 2002), teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão da especificação, do projeto e da codificação.

De acordo com (Myers, 2004), testar é o processo de executar um software com o objetivo de encontrar erros.

Assim sendo, é através do teste de software que se procura aumentar a sua confiabilidade – característica fundamental para a implementação de um bom projeto – e, assim, garantir a sua qualidade.

A importância do software no trabalho diário de diversos segmentos e os custos de atendimento associados com falhas são forças motivadoras para um teste rigoroso e bem planejado. Portanto, a qualidade no desenvolvimento de software é primordial para a sobrevivência das empresas de software. Não é raro uma organização de desenvolvimento de software gastar entre 30 e 40% do esforço total do projeto em testes (Pressman, 2002).

Para testar um software, são necessários bom treinamento, ter experiência, conhecer bem os software e ter criatividade. Mas, é necessário saber que, por mais tempo e energia gastos nos testes, é difícil ter um software livre de erros.

---

[1] Informação obtida no site (http://www.verotthi.com.br/, em 20 de Agosto de 2006) da Empresa Verotthi Consultores, que atua em testes de software e consultoria empresarial.

1.2 Objetivos

O presente trabalho tem o objetivo de definir um processo de teste de software em uma empresa de pequeno porte desenvolvedora de software, com o intuito de obter os seguintes benefícios:

- Melhoria da qualidade dos testes, uma vez que suas atividades são realizadas sistematicamente;
- Redução de esforços para elaboração dos testes, visto que as contribuições para a sua elaboração são apontadas nos artefatos gerados desde o início do desenvolvimento;
- Minimizar os custos associados com a correção de erro identificados, visto que diminui a possibilidade de propagação dos erros entre as diferentes fases do desenvolvimento;
- Transformar o perfil dos desenvolvedores, a fim de que adquiram uma visão mais pragmática e um conhecimento mais profundo das técnicas e das ferramentas para a realização de testes de software passando a ter características de desenvolvedor-testador.

1.3 Metodologia

A metodologia aplicada ao desenvolvimento deste trabalho foi dividida em duas etapas, a saber:

1. Primeira Etapa

Compreendeu a busca de informações e análise dos principais processos, técnicas e estratégias de testes de software.

Durante essa etapa, o trabalho teve um cunho exploratório buscando uma maior familiaridade com o tema e o aprimoramento de idéias. Nesta etapa, a pesquisa bibliográfica foi a abordagem metodológica utilizada.

Conforme Jung (2004), a pesquisa bibliográfica tem por finalidade principal formar uma consistente base “mental” a partir daquilo que é existente. Portanto, proporciona uma ampla aquisição de conhecimentos para o entendimento substancial do assunto,

viabilizando ao pesquisador “ousar” ao propor novos argumentos que justifiquem as descobertas.

A pesquisa bibliográfica utilizou fundamentalmente as contribuições dos diversos autores (Pressman, 2002; Myers, 2004; Sommerville, 2003; Peters & Pedrycs, 2001; Pfleegrer, 1998) sobre Testes de Software, sendo geralmente constituída de material impresso localizado em bibliotecas.

2. Segunda Etapa

Consistiu na definição e no desenvolvimento de um processo de teste de software coerente às necessidades da empresa.

Durante essa etapa do trabalho, foi utilizada a metodologia de pesquisa-ação (Thiollent, 2004). A escolha dessa metodologia foi motivada pelo enfoque dado a este trabalho, onde foi desempenhado um papel ativo no equacionamento do problema considerado como central na pesquisa e no acompanhamento e na avaliação das ações desencadeadas em função dos problemas, atuando para encontrar soluções.

A empresa de pequeno porte está situada no Sul de Minas Gerais e atua no desenvolvimento e na customização de software voltado para as organizações públicas e privadas.

Foram acompanhados e investigados basicamente os seguintes procedimentos:

- Realização de reuniões com a alta direção para a aprovação do trabalho na empresa e a definição dos recursos necessários para o desenvolvimento;
- Realização de diagnóstico da empresa para a elaboração do planejamento das atividades e identificar o estado atual da empresa na área de teste de software;
- Definição e execução do processo de teste.

1.4 Estrutura do Trabalho

Os capítulos subseqüentes desta monografia estão assim organizados:

- No Capítulo 2, são discutidos o processo de teste, a qualidade do processo de teste, as normas em que foram baseados o desenvolvimento deste trabalho e as etapas do processo de teste;
- No Capítulo 3, são apresentadas e discutidas as técnicas e as estratégias de teste de software.
- No Capítulo 4, é apresentada a proposta de processo de teste de software, o SwTest, sendo discutidas suas etapas e apresentados os artefatos gerados, as atividades e os papéis dos membros das equipes de desenvolvimento e teste do software;
- No Capítulo 5, são apresentadas as conclusões, as contribuições e os possíveis desdobramentos deste trabalho.

2 PROCESSO DE TESTE

2.1 Considerações Iniciais

A atividade de teste, quando bem realizada, constitui uma forma de avaliar e agregar qualidade ao software e reduzir custos e trabalho, dando maior confiabilidade ao software desenvolvido.

Para que as empresas de desenvolvimento de software consigam construir seus produtos com qualidade, tornam-se necessárias a elaboração e a utilização de uma estratégia de teste bem definida, pois o processo de teste de software está intimamente ligado com a qualidade de produtos e de processos.

O presente capítulo trata a importância da qualidade, o processo de teste de software, as normas de qualidade ISO/IEC 12119 e IEEE 829 e as etapas de processo de teste.

2.2 Qualidade do Processo de Teste

Atingir um alto nível de qualidade de produto ou de serviço de software é o objetivo da maioria das organizações. Atualmente, não é mais aceitável entregar software com baixa qualidade e reparar os problemas e as deficiências depois de sua entrega ao cliente.

Para descobrir se o software é desenvolvido com qualidade, consumindo o mínimo de recursos e tempo, deve-se considerar que a qualidade do software depende do serviço (resultados) por ele prestado e a facilidade com que se consegue obter esse resultado.

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, dos métodos e das ferramentas empregadas, erros no produto ainda podem ocorrer. Assim, a atividade de teste mostra-se relevante para a identificação e a eliminação de erros que persistem (Maldonado et al, 2003).

Contudo, a qualidade de software é um conceito complexo que não pode ser definido de maneira simples.

A noção de qualidade tem sido de que o produto desenvolvido deve cumprir com sua especificação.

Sommerville (2003) afirma que o teste de software tem o objetivo de garantir que o produto final atenda os requisitos desejados e que seja capaz de tratar corretamente os tipos possíveis de dados de entrada, contribuindo para a garantia de qualidade de software.

Segundo Pressman (2002), é necessário conhecer o que significa qualidade de software para implementar uma estratégia de teste de software, pois ambos estão relacionados. Ao definir a estratégia de testes, define-se também os objetivos da qualidade dos produtos e dos serviços oferecidos.

Uma suposição básica do gerenciamento da qualidade é a qualidade do processo desenvolvido afetar diretamente a qualidade dos produtos fornecidos. A Figura 2.1 ilustra uma abordagem de garantia de qualidade, em que o processo é definido e o produto é desenvolvido de acordo com suas diretrizes, sendo, posteriormente, avaliada a qualidade do produto e do processo usado para seu desenvolvimento.

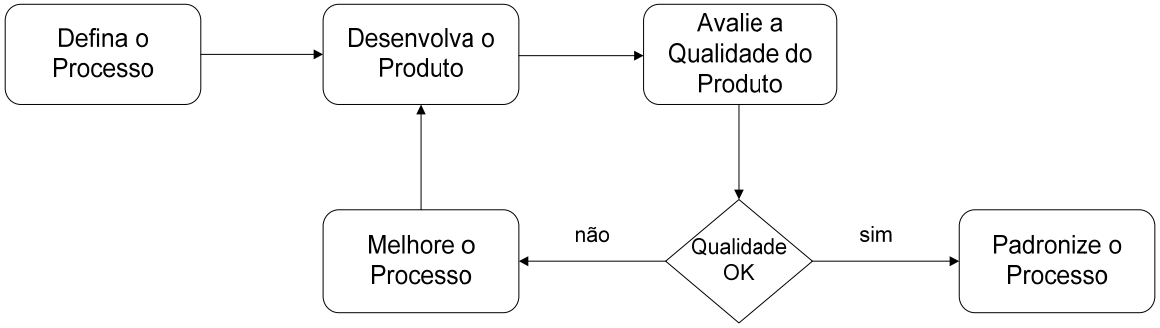


Figura 2.1 - Garantia da Qualidade do Processo (Fonte: (Sommerville, 2003)).

A melhoria da qualidade focaliza a identificação de produtos de boa qualidade, examinando os processos utilizados para desenvolver esses produtos e generalizando esses processos de modo que eles possam ser aplicados em uma série de projetos. Contudo, a relação entre a qualidade do processo de software e os produtos é complexa.

O processo de teste, quando adequadamente definido, pode ter um impacto positivo nos resultados de diversas outras atividades de desenvolvimento. Desta forma, o enfoque das atividades relacionadas ao processo de teste de software não é somente identificar problemas, mas principalmente prevenir problemas.

Sendo assim, é necessário que o processo de teste identifique padrões e procedimentos de teste, tomando como base um processo de desenvolvimento onde, para

cada fase de desenvolvimento, existam atividades relacionadas com o teste, a fim de controlar a qualidade do software desenvolvido.

No desenvolvimento de software, a qualidade do projeto abrange os requisitos, as especificações e o projeto do software.

As atividades de garantia de qualidade de software são coordenadas para dirigir e controlar uma organização, focada em prover confiança de que os requisitos da qualidade serão atendidos.

Os envolvidos no processo de engenharia de software são responsáveis pela qualidade do produto, portanto procedimentos de melhoria de qualidade devem ser implantados e acompanhados pelos responsáveis nas organizações, a fim de (Pressman, 2002):

- Definir o que quer dizer “qualidade de software”;
- Definir um conjunto de atividades que ajudarão a garantir que o trabalho produzido exiba alta qualidade;
- Realizar atividades de garantia de qualidade no projeto de software;
- Usar métricas para desenvolver estratégias para aperfeiçoar o processo de software e, como conseqüência, a qualidade do produto final.

Ao enfatizar a qualidade nas atividades de engenharia de software, por exemplo, no planejamento do processo, análise, definições e testes, é reduzida a quantidade de trabalho que tem que ser refeito devido a defeitos gerados durante essas atividades. Isso resulta em menores custos e, mais importante, menor prazo para colocação no mercado do produto desenvolvido.

Entendido o que é qualidade deve-se então, identificar um conjunto de atividades de garantia de qualidade de software, que filtrarão erros em produtos de trabalho antes que eles sejam passados adiante.

2.3 Controle de Qualidade

O controle de qualidade envolve supervisionar o processo de desenvolvimento de software a fim de assegurar que os procedimentos e os padrões que garantem a qualidade do software desenvolvido sejam seguidos.

O processo de controle de qualidade tem seu próprio conjunto de procedimento e relatórios, que deve ser utilizado durante o desenvolvimento de software. Esses procedimentos devem ser diretos e de fácil compreensão pelas equipes de desenvolvimento e de teste do software.

Há duas abordagens de controle de qualidade: Revisões e Avaliação Automática de Software (Sommerville, 2003).

2.3.1 Revisões

De acordo com Sommerville (2003), as revisões são atividades de verificação da satisfação do processo de software definido com o objetivo de achar erros durante o processo, de modo que eles não se transformem em defeitos depois da entrega do software. Elas envolvem um grupo de pessoas que examina parte ou todo o processo de software, o sistema ou a sua documentação associada, a fim de descobrir possíveis problemas. As conclusões da revisão são formalmente registradas e passadas para o autor ou para quem foi o responsável por corrigir os problemas constatados.

A Tabela 2.1 descreve brevemente os diferentes tipos de revisão.

Tabela 2.1 - Tipos de Revisão (Fonte: (Sommerville, 2003)).

TIPOS DE REVISÃO	PROPÓSITO PRINCIPAL
Inspeção de Projeto ou Programa	Detectar erros detalhados nos requisitos, nos projetos ou no código. A revisão deve ser orientada por um *checklist* de possíveis erros.
Revisões de Progresso	Fornecer informações à gerência sobre o progresso geral do projeto. Essa é uma revisão do processo e de produto, cuja preocupação é com custos, planos e prazos.
Revisões de Qualidade	Realizar uma análise técnica dos componentes ou da documentação do produto, a fim de encontrar inconsistências entre a especificação e o projeto, o código ou a documentação dos componentes e garantir que os padrões de qualidade definidos foram seguidos.

A vantagem da realização das revisões é a descoberta de erros, de modo que eles não se propaguem para o passo seguinte do processo do software.

O propósito da equipe de revisão é detectar erros e inconsistências e mostrá-los ao projetista ou autor do documento. As revisões são baseadas em documentos, mas não se limitam a especificações, projetos ou códigos.

De acordo com Pressman (2002), as revisões de software são um “filtro” para o processo de engenharia de software. Isto é, as revisões são aplicadas em vários pontos durante o desenvolvimento de software e servem para descobrir erros e defeitos que podem ser removidos.

Vários estudos da indústria indicam que as atividades de projeto introduzem entre 50 e 65% dos erros (e em última análise, dos defeitos) durante o processo de desenvolvimento de software. Todavia, foi mostrado que as revisões técnicas formais são 75% efetivas (Jones, 1986) na descoberta de defeitos de projeto. Portanto, a detecção e a remoção de uma alta porcentagem desses erros fazem com que o processo de revisão reduza substancialmente o curso dos passos subseqüentes nas fases de desenvolvimento e manutenção.

2.3.2 Avaliação Automática de Software

De acordo com Sommerville (2003), a avaliação automática consiste no processamento do software e dos documentos produzidos durante seu desenvolvimento por algum programa. Posteriormente, os dados obtidos são comparados com os padrões que se aplicam ao processo de desenvolvimento desse software em particular.

Essa avaliação automática pode envolver avaliação quantitativa de alguns atributos de software. Porém, o uso da medição e de métricas ainda é relativamente incomum, porque os benefícios não são bem definidos (Sommerville, 2003).

2.4 Norma ISO/IEC 12119

A norma ISO/IEC 12119 foi publicada em 1994 e trata da avaliação de pacotes de software (ISO/IEC 12119, 1996). Além de estabelecer os requisitos de qualidade para esse tipo de software, ela também destaca a necessidade de instruções para teste.

Esta norma é aplicável à avaliação de pacotes de software na forma em que são oferecidos e liberados para uso no mercado.

A Figura 2.2 mostra a estrutura básica desta norma. Ela é dividida em requisitos de qualidade e instruções para teste, que serão apresentados a seguir.

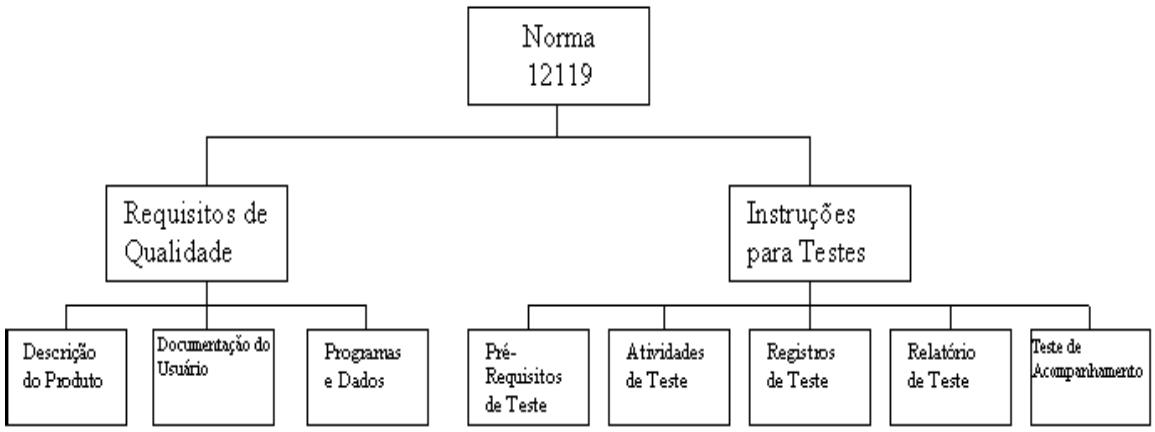


Figura 2.2 - Estrutura da Norma ISO/IEC 12119 (Fonte: (ISO/IEC 12119, 1996)).

2.4.1 Requisitos de Qualidade

Os requisitos de qualidade da norma ISO/IEC 12119 são compostos pela: i) descrição do produto, ii) documentação do usuário; e iii) programas e dados. A seguir, são descritas as principais características dos requisitos:

i) Descrição do Produto

É um documento que expõe as principais propriedades de um pacote de software, com o objetivo de:

- Auxiliar o usuário ou os potenciais compradores deste produto, na avaliação da adequação do produto às necessidades dos usuários;
- Servir como base para testes.

Este documento deve estar disponível ao usuário, independentemente da aquisição do produto, através de um catálogo, de um disquete de apresentação ou de qualquer outro meio disponível que alcance esse objetivo. A descrição deve ser clara, compreensível e

harmônica com os outros documentos associados. A norma propõe aspectos práticos e diretos, indicando “o quê” deve conter esta descrição.

A Tabela 2.2 resume uma parte destas indicações, as quais podem ser Mandatárias (termo “deve”) ou Recomendáveis (termo “pode”). O uso de um requisito como recomendável está diretamente relacionado com o tipo de produto, ou seja, para alguns tipos de produtos esses requisitos podem ser mandatários.

Tabela 2.2 - Requisitos de Qualidade para a Descrição do Produto (Fonte: (ISO12119, 1996)).

ITEM	REQUISITOS
Requisitos Gerais sobre o Conteúdo da Descrição de Produto	O conteúdo da descrição deve ser inteligível e completo e possuir boa organização e apresentação, auxiliando os compradores em potencial na avaliação da adequação do produto às necessidades dos compradores, antes de adquiri-lo.
Identificações e Indicações	Deve apresentar o nome do produto, sua versão, os requisitos de hardware e software, as principais atividades realizadas e os componentes entregues com o pacote.
Declaração sobre Funcionalidade	Deve apresentar uma visão geral das funções disponíveis, os valores limites se existirem e os dispositivos de segurança de acesso ao produto, quando necessários.
Declaração sobre Confiabilidade	Deve apresentar as informações sobre os procedimentos para salvar e recuperar dados.
Declarações sobre Usabilidade	Deve apresentar o tipo de interface com o usuário, se é necessário algum conhecimento técnico específico para o seu uso e se o produto pode ser adaptado às necessidades do usuário.
Declarações sobre Eficiência	Pode incluir informações a respeito do tempo de resposta.
Declarações sobre Manutenibilidade	Pode conter declarações sobre a manutenibilidade do produto.
Declaração sobre Portabilidade	Pode conter declarações sobre a portabilidade do produto.

ii) Documentação do Usuário

É um conjunto completo de documentos, disponíveis na forma impressa ou não, que é fornecido para utilizar um produto de software, sendo também uma parte do produto. Essa documentação deve incluir os dados necessários para a instalação, para o uso da aplicação e para a manutenção do produto de software.

Os principais requisitos da documentação do usuário estão descritos na Tabela 2.3.

Tabela 2.3 - Requisitos de Qualidade para Documentação do Usuário (Fonte: (ISO12119, 1996)).

ITEM	REQUISITOS
Completitude	Deve conter as informações necessárias para o uso do produto de software, tais como estabelecer as funções do pacote, procedimentos de instalação e os valores limites.
Correção	A informação apresentada deve estar correta e sem ambigüidade.
Consistência	Deve haver plena coerência entre a documentação e a descrição do produto. Cada termo deve ter um único significado.
Inteligibilidade	A documentação deve ser compreensível pela classe de usuários que desenvolve atividades com o produto, utilizando termos apropriados, exibições gráficas e explicações detalhadas.
Apresentação e Organização	Deve ser apresentada através de uma forma que facilite uma visão geral, através de índices e tabelas de conteúdo. Se o documento não está na forma impressa, deve haver indicação de como efetuar a impressão.

iii) Programas e Dados

Os requisitos de qualidade para programas e dados utilizam as mesmas definições das características de qualidade da Norma ISO/IEC 9126 (ISO/IEC 9126, 1991). As características de Funcionalidade, Confiabilidade e Usabilidade são destacadas e devem ser verificadas através do uso do produto. Não há requisitos específicos para os aspectos de Eficiência, Manutenibilidade e Portabilidade. Qualquer requisito declarado na documentação do pacote, referente às características citadas, deve estar em conformidade. Os principais requisitos para Programas e Dados estão descritos na Tabela 2.4.

Tabela 2.4 - Requisitos de Qualidade para Programas e Dados (Fonte: (ISO/IEC 12119, 1996)).

ITEM	REQUISITOS
Funcionalidade	Devem ser verificados: o procedimento para a instalação do produto, a presença das funções mencionadas, a execução correta destas funções, a ausência de contradições entre a descrição do produto e a documentação do usuário.
Confiabilidade	O usuário deve manter o controle do produto, sem corromper ou perder dados, mesmo que a capacidade declarada seja explorada até os limites ou fora deles, se uma entrada incorreta é efetuada, ou ainda se instruções explícitas na documentação são violadas.

Tabela 2.4 - Requisitos de Qualidade para Programas e Dados (Fonte: (ISO/IEC 12119, 1996)).(Continuação)

Usabilidade	A comunicação entre o programa e o usuário deve ser de fácil entendimento, através das entradas de dados, mensagens e apresentação dos resultados, utilizando um vocabulário apropriado, representações gráficas e funções de auxílio (help), entre outras; o programa também deve proporcionar apresentação e organização que facilite uma visão geral das informações, além de procedimentos operacionais que auxiliem, por exemplo, a reversão de uma função executada e o uso de recursos de hipertexto em funções de auxílio, entre outras.

2.4.2 Instruções para Testes

Essa norma recomenda como um produto deve ser testado em relação aos requisitos de qualidade. A Tabela 2.5 mostra estas instruções de testes.

Tabela 2.5 - Instruções para Testes (Fonte: (ISO/IEC 12119, 1996)).

FASES	COMPONENTES	RECOMENDAÇÕES
Pré-Requisitos de Testes	Presença de itens necessários ao teste.	Devem estar presentes, para a execução do teste, os componentes a serem entregues e os documentos de requisitos identificados na Descrição do Produto.
	Presença de Componente do Sistema.	Deve estar disponível o ambiente de hardware e software identificados na Descrição do Produto.
	Treinamento	Se o treinamento for mencionado na descrição do produto, o responsável pelo teste deve ter acesso ao material e ao programa de treinamento.
Atividades de Teste	Descrição do Produto	Os requisitos especificados nesta descrição devem ser testados. São instruções detalhadas sobre os procedimentos de teste.
	Documentação do Usuário.	Os requisitos especificados nesta documentação devem ser testados.
	Programas e Dados	Os requisitos especificados para os programas e dados devem ser testados.
Registro de Teste		Os registros devem conter informações suficientes para permitir a repetição do teste, através de um Plano de Teste com os casos de teste, os resultados associados e a identificação das pessoas envolvidas.

Tabela 2.5 - Instruções para Testes (Fonte: (ISO/IEC 12119, 1996)) (Continuação).

Relatório de Teste		Deve conter um resumo com os objetos e os resultados dos testes efetuados com a seguinte estrutura: identificação do produto, sistemas computacionais utilizados, documentos usados, resultados dos testes da Atividade de Teste, uma lista das não conformidades e a data de encerramento do teste.
Teste de Acompanhamento		Quando um produto é testado novamente (considerando o teste anterior), as partes modificadas e as partes inalteradas, mas influenciáveis pelas modificações, devem ser testadas como se fosse um produto novo.

2.5 Norma IEEE 829

A norma IEEE 829 (1998) descreve um conjunto de documentos para as atividades de teste de um produto de software.

Os 8 documentos definidos pela norma, que cobrem as tarefas de planejamento, especificação e relato de testes, são:

1. Plano de Teste – Apresenta o planejamento para execução do teste, incluindo a abrangência, a abordagem, os recursos e o cronograma das atividades de teste. Identifica os itens e a funcionalidade a serem testados, as tarefas a serem realizadas e os riscos associados com a atividade de teste;

2. Especificação de Projeto de Teste – Refina a abordagem apresentada no Plano de Teste identifica a funcionalidade e as características a serem testadas pelo projeto e por seus testes associados. Este documento também identifica os casos e os procedimentos de teste, se existirem, e apresenta os critérios de aprovação;

3. Especificação de Caso de Teste – Define os casos de teste, incluindo dados de entrada, resultados esperados, ações e condições gerais para a execução do teste;

4. Especificação de Procedimento de Teste – Especifica os passos para executar um conjunto de casos de teste;

5. Diário de Teste – Apresenta registros cronológicos dos detalhes relevantes relacionados com a execução dos testes;

6. Relatório de Incidente de Teste – Documenta qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior;

7. Relatório-Resumo de Teste – Apresenta de forma resumida os resultados das atividades de teste associados com uma ou mais especificações de projeto de teste e provê avaliações baseadas nesses resultados;

8. Relatório de Encaminhamento de Item de Teste – Identifica os itens encaminhados para teste no caso de equipes distintas serem responsáveis pelas tarefas de desenvolvimento e de teste.

Os documentos, Especificação de Projeto de Teste, Especificação de Caso de Teste e Especificação de Procedimento de Teste, cobrem a tarefa de especificação de testes. Os documentos, Diário de Teste, Relatório de Incidente de Teste, Relatório-Resumo de Teste e Relatório de Encaminhamento de Item de Teste, cobrem a tarefa de relato de teste.

2.6 Etapas do Processo de Teste

O Processo de Teste é caracterizado por um conjunto de atividades que são executadas ao longo de todo o ciclo de desenvolvimento do software (Villas, 2003). Essas atividades estão agrupadas em etapas bem definidas, a saber:

1. Etapa de planejamento

Planejar é a distribuição racional no tempo dos recursos disponíveis para a realização de alguma atividade. De modo geral, nesta etapa, decide-se antecipadamente o que deve ser feito e quando deve ser feito, ou seja, traça uma linha de ação. É justamente nesta etapa que será descrito o escopo dos testes, identificando métodos que serão empregados, recursos necessários, cronograma de atividades, pessoal necessário, itens que serão testados, itens que não serão testados, características dos itens testados e responsabilidades;

2. Etapa de projeto

Projetar é especificar de maneira detalhada a forma de realizar algo. Nesta etapa, será escolhido um grupo específico de características a serem testadas, descrevendo detalhadamente os métodos que serão empregados e os testes que deverão ser feitos;

3. Etapa de implementação

Implementar é fornecer os subsídios indispensáveis à execução de alguma tarefa. Na etapa de implementação, são descritos detalhadamente os itens que serão testados, as entradas utilizadas, as saídas esperadas e os passos necessários para executar os testes, ou seja, a seqüência de tarefas necessária para analisar um software com a finalidade de avaliar um conjunto de suas características;

4. Etapa de execução

Executar é a parte prática do teste, isto é, aquela em que se efetiva a tarefa de teste propriamente dita; a partir do procedimento de teste que será seguido e dos casos de teste associados, o testador realizará os testes. Nesta etapa, ele deve registrar toda a atividade de teste, identificando data e hora do teste, autor do teste, o procedimento seguido, o pessoal envolvido, os resultados obtidos, as condições ambientais e os eventos não esperados (quando ocorrerem). Caso aconteçam problemas durante os testes, estes deverão ser relatados, sendo referidos o procedimento em uso, os casos de teste associados e os itens que estão sendo testados, e fornecendo uma descrição dos problemas, as entradas utilizadas, a data e a hora do ocorrido, o passo do procedimento, os observadores e o impacto do problema sobre os testes. Ao final das atividades de teste, um resumo deve ser fornecido referindo os itens testados, sumarizando os resultados obtidos e fornecendo uma avaliação baseada nesses resultados.

Embora essas etapas ocorram em seqüência, nem sempre elas acontecem em bloco para cada teste; na verdade, elas são diluídas ao longo do ciclo de vida do projeto.

Após a fase de Especificação de Requisitos, a partir do documento onde estão especificados os requisitos do sistema a ser construído, um Plano de Teste deve ser gerado. Com base nesse plano, são gerados os Projetos de Teste e os Casos de Teste.

Com base no documento de Especificação de Características, os Planos, os Projetos e os Casos de Teste devem ser gerados. Baseado no documento de Especificação de Definição das Interfaces, os Planos, os Projetos e os Casos de Teste de Integração são gerados.

Na fase de Implementação, são gerados os Procedimentos de Teste. Neste momento, cada programador, ao finalizar a implementação do software que está sob sua

responsabilidade, executa o teste de unidade para seu software. A partir de grupos de unidades corretas, são iniciados os testes de unidade. Caso seja encontrado algum problema nesse teste, as unidades que apresentaram defeito são devolvidas para a equipe de desenvolvimento (com os relatórios de problemas) para serem corrigidas e testadas novamente (teste de unidade).

Quando um grupo de requisitos que compõem um software passa pelo teste de unidade, pode ser iniciado o teste de integração (que, em caso de problema, sofre o mesmo procedimento descrito anteriormente). Quando os testes de integração terminam, pode-se executar o teste de sistema.

2.7 Considerações Finais

As atividades realizadas durante a execução de testes são consideradas geralmente, por diferentes autores, como bastante dispendiosas em relação às demais fases do desenvolvimento de um sistema (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001). Esta afirmação pode ser justificada por diferentes fatores, tais como:

- Realização de testes sem planejamento: a fase de testes tende a ser longa e imprevisível;
- Não formalização da arquitetura do sistema: mais esforço será despendido durante a identificação dos testes necessários, uma vez que as restrições existentes no sistema não foram claramente especificadas;
- Não reprodução do teste: dados de entrada não documentados implicam em não uniformidade da verificação e retrabalho;
- Falta de histórico de explicitar as atividades relacionadas com o teste: estas atividades pesam na estimativa de tempo e de recurso e, conseqüentemente, quando consideradas nos cronogramas, surpreendem negativamente o gerente e o cliente.

Sendo assim, o presente capítulo apresentou as atividades relacionadas com o teste, mostrando que é necessário elas serem identificadas de forma realística, tendo como base um processo de desenvolvimento onde, para cada fase do desenvolvimento, existam atividades relacionadas com o teste.

3 TESTE DE SOFTWARE

3.1 Considerações Iniciais

Segundo Myers (2004), teste é um processo de execução de um programa com a finalidade de encontrar erros. Um bom caso de teste é aquele que tem alta probabilidade de encontrar erros ainda não descobertos. Se o teste for conduzido de maneira bem-sucedida, ele descobrirá erros de software.

Um erro pode ser o resultado de:

- A especificação pode estar errada ou ter a falta um requisito;
- A especificação pode conter um requisito impossível de implementar, considerando o hardware e o software estabelecidos;
- O projeto do sistema pode conter um defeito (banco de dados e linguagem de programação);
- O projeto do programa pode conter um defeito ou o código do programa pode estar errado.

Os princípios de testes de software, segundo Pressman (2002), são:

- Todos os testes devem ser relacionados aos requisitos do cliente;
- Os testes devem ser planejados muito antes do início do teste;
- O principio de Pareto[1] se aplica aos testes de software, isto é, 80% dos erros descobertos durante o teste vão provavelmente ser relacionado a 20% dos componentes do software. O problema é isolar os componentes suspeitos e testá-los;

---

[1] O Princípio de Pareto foi criado no Séc. XIX pelo economista italiano chamado Alfredo Pareto que, ao analisar a sociedade concluiu que grande parte da riqueza se encontrava nas mãos de um número demasiado reduzido de pessoas. Após concluir que este princípio estava válido em muitas áreas da vida quotidiana, estabeleceu o designado método de análise de Pareto, também conhecido como dos 20-80% e que significa que um pequeno número de causas (geralmente 20%) é responsável pela maioria dos problemas (geralmente 80%).(Fonte: http://www.notapositiva.com/dicionario_gestao/principio_pareto.htm, 19 de Agosto de 2006).

- O teste deve começar “no varejo”, isto é, nos componentes individuais, para depois progredir até o teste “no atacado”, em todo o software;
- Teste completo não é possível, mas pode-se cobrir adequadamente a lógica do software e garantir que as condições no projeto ao nível de componente tenham sido exercitadas.

Um bom teste tem uma alta probabilidade de encontrar erro, não deve ser redundante, muito simples e muito complexo (Myers, 2004).

3.2 Técnicas de Teste de Software

O projeto de testes para software e outros produtos que passam por engenharia pode ser tão desafiante quanto o projeto inicial do produto propriamente dito.

Para realizar testes em um software, é necessário definir casos de teste, com o intuito de exercitar as diferentes características de um software.

Na literatura, existem diversas técnicas de teste de software, as quais podem ser divididas nas seguintes categorias: caixa-branca, caixa-preta, orientada a objetos e baseada em estados (Pressman, 2002; Sommerville, 2003; Myers, 2004; Pfleeger, 1998Peters & Pedrycz, 2001).

Basicamente, estas técnicas podem diferir na abordagem usada para avaliar ou construir os conjuntos de casos de teste, contribuindo para que diferentes características do software sejam testadas adequadamente.

As subseções a seguir apresentam uma breve descrição das técnicas de teste de software citadas anteriormente.

3.2.1 Teste Caixa-Branca

O objetivo dos testes caixa-branca é exercitar partes específicas do código do sistema, para garantir que a parte interna dos módulos seja adequadamente testada. Para tanto, a estrutura interna do código-fonte deve ser conhecida.

Pressman (2002) afirma que, utilizando esses testes, o engenheiro de software pode derivar os casos de teste de forma que:

- Garantam que os caminhos independentes dentro de um módulo tenham sido exercitados pelo menos uma vez;
- Exercitem as decisões lógicas para valores *booleanos*;
- Executem os laços em sua fronteira e dentro de seus limites operacionais;
- Exercitem as estruturas de dados internas para garantir a sua validade.

A limitação apresentada pelo teste de caixa de branca é o fato de não permitir confirmar que os requisitos do usuário foram atendidos, visto que apenas os módulos específicos são testados.

Como exemplos, têm-se as seguintes técnicas de teste caixa-branca (Sommerville, 2003; Pressman, 2002; Peters & Pedrycs, 2001):

- **Teste de Caminho:** permite que as linhas de código do software sejam testadas pelo menos uma vez, embora não possa garantir que as combinações lógicas do código tenham sido testadas;
- **Teste de Condição:** permite testar as condições lógicas presentes em um módulo do software, considerando as relações com operadores relacionais e *booleanos*;
- **Teste de Laços:** consiste em testar as estruturas de repetição definidas para o software sendo avaliado, analisando os laços em suas fronteiras e dentro de seus limites operacionais.

3.2.2 Teste Caixa-Preta

Os testes caixa-preta focalizam os requisitos funcionais do software, tentando encontrar erros das seguintes categorias: funções incorretas ou omitidas; erros de interface; erro de estrutura de dados ou de acesso à base de dados externa; erros de comportamento ou desempenho e erros de iniciação e término.

É uma abordagem complementar que deve ser usada após os testes de caixa branca. O objetivo é mostrar que as funções do software estão operacionais, as entradas são adequadamente aceitas e as saídas são corretamente produzidas e a integridade da informação externa, como, por exemplo, um arquivo, é mantida. Os casos de teste devem

ser gerados com base no comportamento externamente visível, sem se basear na sua implementação.

O teste de caixa-preta tende a ser realizado durante os últimos estágios do teste, procurando satisfazer as questões (Pressman, 2002):

- Como a validade funcional é testada;
- Como o comportamento e o desempenho do sistema são testados;
- Que classes de entrada vão construir bons casos de teste;
- O sistema é particularmente sensível a certos valores de entrada;
- Como são isolados os limites de uma classe de dados;
- Que taxas e volumes de dados o sistema pode tolerar;
- Que efeito as combinações específicas de dados vão ter na operação do sistema.

Como exemplos, têm-se as seguintes técnicas de teste caixa-preta (Sommerville, 2003; Pressman, 2002; Peters & Pedrycs, 2001):

- **Particionamento de Equivalência:** permite encontrar classes de erros, com base na divisão do domínio de entradas em classes de dados, contribuindo para a redução do número de casos de teste que precisam ser desenvolvidos;
- **Análise de Valor Limite:** permite avaliar se o código proposto é capaz de manipular situações de exceção, ou seja, se o código trata corretamente as informações de entrada que se encontram nas fronteiras das classes de equivalência definidas.
- As restrições identificadas para o sistema de software deverão ser exercitadas durante a realização dos testes para analisar a reação do sistema;
- **Técnicas de Grafo Causa-Efeito:** permite a avaliação de conjuntos complexos de condições de entrada (causa) e ações (efeito), de forma a definir um grafo causa-efeito para, em seguida, convertê-lo em uma tabela de decisão de onde serão extraídos os casos de teste.

3.2.3 Teste Orientado a Objetos

De acordo com Pressman (2002), o objetivo do teste é encontrar o maior número possível de erros, com um mínimo de esforço aplicado, durante um intervalo de tempo realístico. Apesar deste objetivo permanecer o mesmo para software orientado a objetos, a natureza do software orientado a objeto muda tanto a estratégia, como a tática de testes.

A arquitetura de software orientado a objetos resulta em uma série de unidades de software em camadas que encapsulam as classes colaboradoras. É necessário um software orientado a objetos em uma variedade de níveis diferentes, em um esforço para descobrir erros que podem ocorrer à medida que as classes colaboram umas com as outras e as unidades do software se comunicam entre as camadas arquiteturais.

O teste orientado a objetos é realizado pelos engenheiros de software e especialistas em teste e é estrategicamente semelhante ao testes de software convencionais, mas é taticamente diferente. Uma vez gerado o código, o teste orientado a objetos começa em pequena escala com o teste de classe. Uma série de testes é projetada para exercitar as operações das classes e examinar se existem erros quando uma classe colabora com outras classes. À medida que as classes são integradas para formar um software, são aplicados testes baseados no caminho de execução, apoiado no uso em agrupamento, junto com abordagens fundamentadas em erros, para exercitar plenamente as classes colaboradoras. Finalmente, são usados casos de uso para descobrir erros no nível de validação do software.

De acordo com Myers (2004), a construção de software orientado a objetos começa com a criação de modelos de análise e de projeto. Devido à natureza evolutiva do paradigma de engenharia de software orientado a objeto, esses modelos começam como representações relativamente informais dos requisitos do software e evoluem para modelos detalhados de classes, conexões e relações de classes, projetos e alocação do software e projeto de objetos.

A notação e a sintaxe usadas para representar os modelos de análise e de projeto estão ligadas ao método específico escolhido para o desenvolvimento do software. Assim sendo, a correção sintática é julgada quanto ao uso adequado da simbologia; cada modelo é revisado para garantir que as convenções adequadas de modelagem foram mantidas.

Em aplicações convencionais, o teste de unidade está voltado para a menor unidade compilável do software – uma função ou procedimento. Após, elas são integradas em uma estrutura de programa, enquanto uma série de testes de regressão[1] é executada para descobrir erros devido à interligação entre os módulos e a efeitos colaterais causados pela adição de novas unidades. Finalmente, o sistema todo é testado para garantir que os erros nos requisitos são descobertos.

No software orientado a objeto, o conceito de unidade se modifica, devido ao encapsulamento que guia a definição de classes e objetos. Cada classe e cada instância de uma classe (objeto) empacotam os atributos (dados) e as operações que manipulam esses atributos. Ao invés de testar um módulo individual, a menor unidade testável é a classe ou o objeto encapsulado. Não se pode testar uma única operação isoladamente, mas como parte de uma classe.

No software orientado a funções, existe uma distinção entre as unidades básicas de software (funções) e os conjuntos dessas unidades de software (módulos). Em software orientado a objetos, não há tal distinção.

As estratégias de integração do software orientado a funções são muitas vezes inadequadas ao orientado a objetos.

Como na validação convencional, a validação do software orientado a objeto focaliza ações usuário-notado e a saída do sistema usuário-reconhecido. Para ajudar na derivação dos testes de validação, o testador deve apoiar-se nos casos de uso.

Os métodos de projeto de casos de teste para software orientado a objetos estão ainda em evolução. Este teste está voltado para o projeto de seqüências apropriadas de operações, para exercitar os estados de uma classe (Myers, 2004).

A classe de software orientado a objetos é o alvo do projeto de casos de testes. Como os atributos e as operações são encapsulados, o teste de operações fora da classe é geralmente improdutivo.

---

[1] Reexecução de algum subconjunto de testes que foram conduzidos para garantir que as modificações não propagaram efeitos colaterais indesejáveis.

A estrutura concisa de muitas operações de classes faz com que diversos autores (Sommerville, 2003; Peters & Pedrycs, 2001; Pfleeger, 1998; Pressman, 2002) argumentem que o esforço aplicado ao teste caixa-branca poderia ser mais bem redirecionado para testes no nível de classes.

Os métodos de teste caixa-preta são tão adequados para software orientado a objetos quanto são para sistemas desenvolvidos usando os métodos convencionais de engenharia de software.

3.2.4 Teste Baseado em Estados

Esta técnica é utilizada em sistemas que possuem estados significativos durante a sua execução.

O objetivo do teste baseado em estados é validar o comportamento do sistema ou unidade sendo testada. Para tanto, os casos de teste desenvolvidos deverão representar as seqüências de eventos que acarretam as mudanças de estados esperadas.

Como exemplo, tem-se a técnica de teste baseada em estados denominada W, a qual permite a geração de seqüências de teste baseadas na máquina de estados finitos que modela o sistema (Souza, 2003).

3.3 Estratégias de Teste de Software

Na área da engenharia de software, as estratégias de teste de software proporcionam os detalhes de como fazer para construir um programa. Elas envolvem um amplo conjunto de tarefas que incluem: planejamento, análise de requisitos, projeto da estrutura de dados, arquitetura de programas e algoritmo de processamento, teste e manutenção (Pressman, 2002).

Uma estratégia de teste de software integra métodos de projeto de casos de teste em uma série bem-planejada de passos, que resultam na construção de software bem-sucedido. A estratégia deve ser flexível a fim de promover uma abordagem de teste sob medida e também rígida para promover planejamento razoável e acompanhamento gerencial, à medida que o projeto progride.

O teste freqüentemente responde por mais esforço de projeto que qualquer outra atividade de engenharia de software e começa no varejo e progride para o atacado, isto é, de um programa para o sistema como um todo (Pressman, 2002).

Uma especificação de teste documenta a abordagem da equipe de software para o teste, definindo um plano que descreve uma estratégia global e um procedimento que define passos específicos de teste e os testes que serão conduzidos (Sommerville, 2003).

Um plano e um procedimento de testes efetivos vão levar à construção coordenada do software e à descoberta de erros em cada estágio do processo de construção.

Um conjunto de passos no qual se pode colocar técnicas e projetos de casos de testes e métodos de teste específicos deve ser definido para o processo de testes.

As características de um conjunto de testes são (Pressman, 2002):

- O teste começa em nível de componente e prossegue de acordo com as etapas de integração entre os componentes;
- Diferentes técnicas de testes são adequadas em diferentes momentos;
- O teste é conduzido pelo desenvolvedor do software e um grupo de testes;
- O teste e a depuração são atividades diferentes, mas a depuração deve ser acomodada em qualquer estratégia de testes.

A Figura 3.1 mostra a fase em que as estratégias de teste, especificadas adiante, são aplicadas.

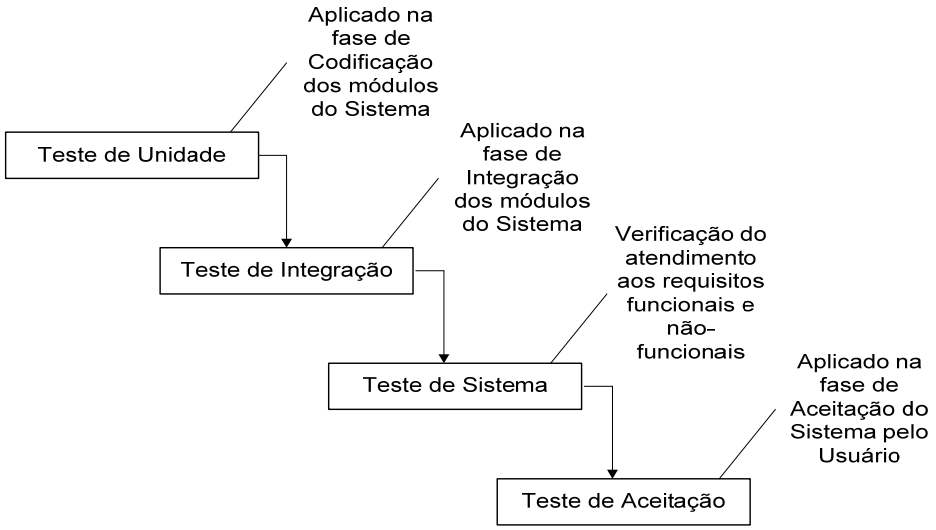


Figura 3.1 - Estratégias de Teste (Elaborado pelo Autor).

3.3.1 Teste de Unidade

O teste de unidade é o primeiro e mais básico teste que o software de um projeto pode oferecer (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001; Myers, 2004). Deve-se lembrar que unidade é entendida como sendo um componente indivisível de código sob a responsabilidade de apenas um programador.

O teste de unidade focaliza o esforço de verificação na menor unidade de projeto de software – o componente ou módulo de software. Tem a finalidade de verificar separadamente se cada módulo do software foi corretamente implementado. Tais módulos correspondem aos procedimentos e às funções definidas para o sistema de software.

Em geral, esse teste é dividido em duas fases. A primeira se concentra em depurar a unidade em seu contexto léxico-sintático (por exemplo, por meio do uso de compiladores); nesse ponto, o uso de listas de verificação (*check-list*) e inspeção visual do código também traz bons resultado. A segunda tem o objetivo de verificar a lógica da unidade. Nessa fase, a proposta é exercitar as interfaces, os comandos iterativos, os comandos condicionais, a manipulação das estruturas de dados e a verificação do fluxo de controle.

A importância do teste de unidade, portanto, se dá ao fato de que, quando os módulos não são unitariamente testados, o teste de integração pode sofrer grandes atrasos devido às paralisações para correções de módulos individuais. Normalmente, os testes de

unidade são realizados pelo próprio programador, em virtude dos conhecimentos que possui em relação ao código gerado.

3.3.2 Teste de Integração

O teste de integração é uma técnica sistemática para construir a estrutura do software, enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces. O objetivo é tomar componentes testados em nível de unidade e construir a estrutura do software determinada pelo projeto. Estes testes devem ser desenvolvidos a partir da especificação do sistema e começar logo após que alguns dos componentes do sistema estejam disponíveis (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001; Myers, 2004).

O foco principal é o teste da comunicação entre os módulos, garantindo que eles funcionam adequadamente juntos.

A integração deve ser realizada de forma incremental, ou seja, integrar os módulos de forma gradual, para facilitar a identificação e a correção dos erros. Para realizar os testes de integração dos módulos, existem duas estratégias definidas (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001; Myers, 2004):

- **Integração *Top-Down*:** é uma abordagem incremental para a construção da estrutura de um programa. Os módulos são integrados movendo-se descendentemente pela hierarquia de controle, começando com o módulo de controle principal são incorporados à estrutura de maneira primeiro-em-profundidade ou primeiro-em-largura;

- **Integração *Bottom-Up*:** os componentes de nível inferior são integrados e testados antes que os componentes de nível superior tenham sido desenvolvidos. Representa uma alternativa para testar a restrição apresentada para a utilização da estratégia *top-down*. Neste caso, os testes de integração devem ser iniciados considerando aqueles módulos localizados no nível mais baixo da hierarquia de controle, seguindo as dependências de uso. Na estratégia *bottom-up*, a visão global do sistema é obtida após a integração dos módulos (Pressman, 2002), sendo recomendado o seu uso

para sistemas de grande porte, onde cada equipe de desenvolvimento é responsável pelo desenvolvimento de um subsistema.

A escolha entre as estratégias deve ser feita com base nas características do software considerado, com o intuito de selecionar aquela que melhor atende às necessidades do sistema em desenvolvimento. Estas estratégias podem ser combinadas, de forma que atendam às reais características definidas na implementação.

3.3.3 Teste de Sistema

O teste de sistema tem a finalidade de verificar a adequação do software desenvolvido interagindo com outros elementos do sistema (hardware definitivo, rede, informações, etc.). Isso é necessário, pois de acordo com Pressman (2002), o software é apenas um elemento de um sistema baseado em computador.

Com a realização do teste de sistema, a equipe de desenvolvimento pode garantir que o sistema está pronto para o teste de aceitação do cliente. Com isso, é possível verificar que a tecnologia foi usada adequadamente e, quando as partes componentes do sistema são agrupadas, elas funcionam corretamente.

As seguintes atividades são consideradas no teste de sistema (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001; Myers, 2004):

- **Teste de Recuperação:** verifica a recuperação do sistema, através da inserção de falhas no software de diferentes maneiras. Se a recuperação é automática, sendo realizada pelo próprio sistema, a reinicialização, os mecanismos de verificação e a recuperação dos dados devem ser avaliados. Se a recuperação requer intervenção humana, o tempo de correção deve ser avaliado para determinar se está dentro dos limites aceitáveis;

- **Teste de Segurança:** verifica se os mecanismos de proteção construídos para o sistema são capazes de protegê-lo de invasões. Geralmente, o objetivo é fazer o custo da invasão se tornar mais alto do que o valor da informação que o invasor irá obter;

- **Teste de Sobrecarga:** determina como o sistema se comporta diante de situações não usuais e/ou de sobrecarga. Tais situações podem ser número grande de usuários, dados transmitidos, relatórios impressos, etc;

- **Teste de Desempenho:** verifica o desempenho do software em relação a tempo de resposta, uso de processador ou de memória, acesso aos dados, etc., durante a sua execução. Em sistemas de tempo real, por exemplo, o software que produz a funcionalidade esperada, mas tem o tempo de processamento maior do que o especificado é inaceitável;

- **Teste de Instalação:** verifica se o manual de instalação está adequado e se a aplicação pode ser instalada com sucesso, considerando a memória e o espaço em disco disponíveis, a existência de programas de infra-estrutura necessários (por exemplo, dll), identificação de possíveis aplicações que serão afetadas pela sua instalação, etc.

3.3.4 Teste de Aceitação do Sistema

O teste de aceitação deve ser realizado pelo cliente quando partes significativas ou o sistema de software como um todo foram consideradas concluídas pela equipe de desenvolvimento, após a realização do teste de sistema.

É importante salientar que o teste de sistema deve ser executado, primeiramente, pelo projetista. Entretanto, após o teste ter sido concluído, é necessário que o sistema seja testado pelo cliente. O teste de sistema realizado pelo cliente pode ser classificado como teste de aceitação, teste *alpha* e teste *beta* (Pressman, 2002; Sommerville, 2003; Pfleeger, 2001; Myers, 2004).

O objetivo é evitar surpresas desagradáveis depois que o sistema entra em funcionamento. Neste tipo de teste, é realizada a comparação entre os requisitos iniciais e as necessidades dos usuários finais. O teste de aceitação determina se a implementação está ou não em conformidade com o documento alvo (descrição de um sistema a ser implementado, que, neste caso, é construído na fase de análise de requisitos). Pequenas correções ainda podem ser feitas para ajustar características de sistema.

O teste de aceitação não avalia o trabalho dos programadores, mas do analista. Pode ser (Souza, 2003):

- **Teste *Alpha*:** é um teste realizado por um usuário final em um ambiente controlado, onde o projetista observa as reações do usuário e registra os erros e os problemas. O objetivo é permitir que o usuário manipule o software ainda não concluído, buscando identificar possíveis problemas;

- **Teste *Beta*:** é um teste realizado após o teste *alpha*, o projetista não está presente. Os testes são conduzidos pelos usuários finais do software que registra os problemas encontrados, relatando ao projetista.

3.3.5 Considerações Finais

De acordo com Peters & Pedrycz (2001), a questão do teste de software precisa ser totalmente considerada em qualquer sistema de software. É preciso estar totalmente ciente das implicações da utilização de metodologia específica de teste, da intensidade da abrangência apoiada por um determinado método de teste, dos custos associados ao procedimento de teste e de uma necessidade de uma combinação racional entre esses aspectos.

4 SWTEST – UM PROCESSO DE TESTE DE SOFTWARE

4.1 Considerações Iniciais

O processo de teste de software SwTest propõe a organização das atividades de teste de software em uma empresa de pequeno porte desenvolvedora de software, a fim de solucionar os seguintes problemas:

- Falta de planejamento dos tempos e dos custos dos testes de software;
- Falta de documentação, pois os testes não são devidamente preparados, não há um plano de teste e os testes não são documentados;
- O teste é a última etapa do desenvolvimento, ou seja, é a última atividade com que o gerente e a equipe de desenvolvimento do projeto se preocupam em realizar;
- Os casos de testes são gerados de forma aleatória, ou seja, os testes são escolhidos com base no conhecimento dos testadores, que não possuíam experiência e sem procedimento definido;
- Grande quantidade de retrabalho, devido a falta de planejamento dos testes e detecção de erros, o que acabava por acarretar mais custos e atraso na entrega dos projetos da Empresa.

Tendo em vista os problemas apresentados, foi definido o Software Teste (SwTest).

Esta seção apresenta a concepção do processo, suas atividades, as ferramentas utilizadas, os artefatos gerados e os papéis desempenhados pelas pessoas envolvidas no processo.

4.2 Concepção do Processo SwTest

Na fase de concepção do processo SwTest, foi elaborado um plano de ação com base nos resultados encontrados no diagnóstico de problemas da empresa anteriormente apresentados, a fim de uma melhor condução das atividades relacionadas ao processo de teste.

O plano de ação identifica os objetivos do projeto, quais os seus impactos sobre a organização, o método de trabalho que será utilizado, as ações específicas e os recursos necessários para a definição e a implantação do processo de teste de software.

Assim, nesta fase, foi gerado um documento contendo as ações que deveriam ser executas dentro do propósito de definir e implantar o processo de teste de software da organização de acordo com o diagnóstico realizado.

Após o término do plano de ação e de estudos sobre processo, qualidade, técnicas e estratégias de teste de software o processo SwTest foi elaborado e detalhado.

O processo SwTest foi organizado em atividades, onde cada uma possui um conjunto de entradas e saídas necessárias para a sua realização, as tarefas a serem realizadas e as responsabilidades.

O SwTest foi definido usando o modelo de processo de desenvolvimento de software existente na empresa, para que sua implantação seja menos traumática para os funcionários e para a gerência da empresa.

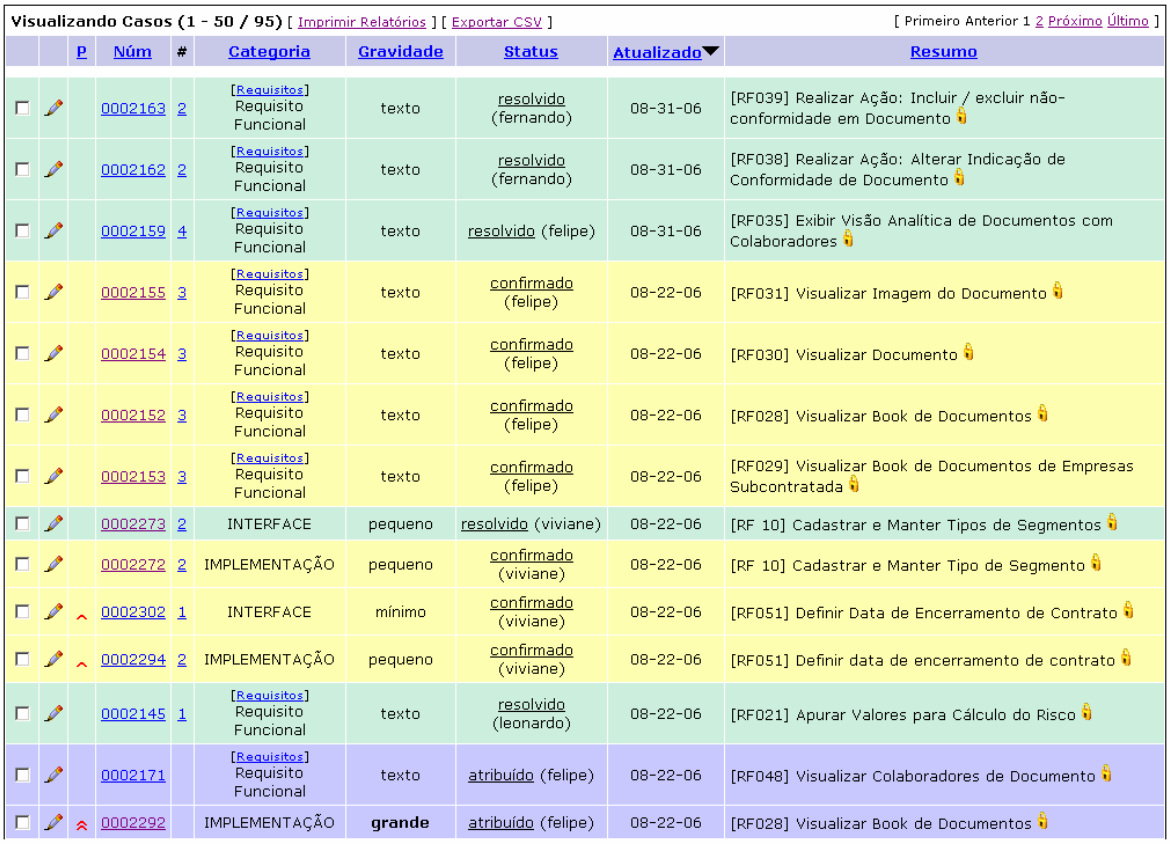
Para viabilizar a implantação, foi utilizado o sistema Mantis. O Mantis é um sistema de rastreamento e acompanhamento de mudanças (*tracking systems*), contextualizado principalmente em termos de gerência de modificações pedidas e efetuadas no software, como também no acompanhamento de erros[1] encontrados e corrigidos. Assim sendo, ele é utilizado para manutenção e evolução do software desenvolvido, principalmente pelo fato de permitir o acompanhamento da evolução de um pedido de alteração no software. A Figura 4.1 apresenta a página de visualização de registro de casos no Mantis

Este sistema também pode ser utilizado em atividades de revisões da documentação do software, na medida em que permitem que arquivos sejam anexados, facilitando o acesso a artefatos em análise.

O Mantis é adequado para manter um histórico e a evolução da resolução de incidentes[2] em projetos. Os usuários podem inserir novos incidentes, ver os incidentes,

[1] No contexto do presente trabalho, é a manifestação de um incidente.
[2] Incidentes, no contexto do presente trabalho, são problemas de especificação, projeto ou implementação do software.

pois o Mantis é um sistema adequado para manter o histórico e a evolução da resolução de incidentes em projetos. Os usuários podem inserir novos incidentes, ver os incidentes por resolver, os incidentes resolvidos, a documentação associada à resolução do incidente e, por fim, analisar os dados estatísticos gerados automaticamente pelo sistema. Além de gerir os dados sobre os incidentes, de forma simples e prática, o Mantis alerta (notifica) os usuários através de mensagens de correio eletrônico a fim de informá-los das evoluções recentemente sofridas pelo software e dos incidentes e erros detectados no processo de desenvolvimento do software.



Visualizando Casos (1 - 50 / 95) [Imprimir Relatórios] [Exportar CSV] [Primeiro Anterior 1 2 Próximo Último]

		P	Núm	#	Categoria	Gravidade	Status	Atualizado	Resumo
			0002163	2	[Requisitos] Requisito Funcional	texto	resolvido (fernando)	08-31-06	[RF039] Realizar Ação: Incluir / excluir não-conformidade em Documento
			0002162	2	[Requisitos] Requisito Funcional	texto	resolvido (fernando)	08-31-06	[RF038] Realizar Ação: Alterar Indicação de Conformidade de Documento
			0002159	4	[Requisitos] Requisito Funcional	texto	resolvido (felipe)	08-31-06	[RF035] Exibir Visão Analítica de Documentos com Colaboradores
			0002155	3	[Requisitos] Requisito Funcional	texto	confirmado (felipe)	08-22-06	[RF031] Visualizar Imagem do Documento
			0002154	3	[Requisitos] Requisito Funcional	texto	confirmado (felipe)	08-22-06	[RF030] Visualizar Documento
			0002152	3	[Requisitos] Requisito Funcional	texto	confirmado (felipe)	08-22-06	[RF028] Visualizar Book de Documentos
			0002153	3	[Requisitos] Requisito Funcional	texto	confirmado (felipe)	08-22-06	[RF029] Visualizar Book de Documentos de Empresas Subcontratada
			0002273	2	INTERFACE	pequeno	resolvido (viviane)	08-22-06	[RF 10] Cadastrar e Manter Tipos de Segmentos
			0002272	2	IMPLEMENTAÇÃO	pequeno	confirmado (viviane)	08-22-06	[RF 10] Cadastrar e Manter Tipo de Segmento
		^	0002302	1	INTERFACE	mínimo	confirmado (viviane)	08-22-06	[RF051] Definir Data de Encerramento de Contrato
		^	0002294	2	IMPLEMENTAÇÃO	pequeno	confirmado (viviane)	08-22-06	[RF051] Definir data de encerramento de contrato
			0002145	1	[Requisitos] Requisito Funcional	texto	resolvido (leonardo)	08-22-06	[RF021] Apurar Valores para Cálculo do Risco
			0002171		[Requisitos] Requisito Funcional	texto	atribuído (felipe)	08-22-06	[RF048] Visualizar Colaboradores de Documento
		^^	0002292		IMPLEMENTAÇÃO	**grande**	atribuído (felipe)	08-22-06	[RF028] Visualizar Book de Documentos

Figura 4.1 - Página de Visualização dos Casos Relatados no Mantis (Fonte: A Empresa)

O Mantis foi escolhido por seu uso ser institucionalizado na empresa, o que facilitou a parte de registro e relato de incidentes detectados nas atividades de revisão e de erros encontrados durante a fase de execução de teste.

Porém, foi necessário definir um padrão de uso da ferramenta, pois, para cada projeto, eram adotados uma configuração e um fluxo de *status*.

Fluxo de *status* é a representação do estado atual do incidente e/ou erro relatado. Classificam-se em:

- NOVO: um novo relatório foi gerado;
- ATRIBUÍDO: o novo relatório gerado foi analisado e encaminhado para o usuário responsável pelo seu tratamento e/ou sua correção;
- ADMITIDO: o relatório foi aceito pelo usuário para o qual ele foi atribuído;
- RESOLVIDO: o incidente e/ou erro informado no relatório foi solucionado e deve ser revisado e/ou analisado pelo usuário responsável por sua concepção;
- RETORNO: o novo relatório gerado foi analisado, porém o usuário responsável por sua análise não concorda com sua descrição encaminhando ao usuário responsável por sua criação para melhorá-lo ou removê-lo do projeto;
- CONFIRMADO: o relatório resolvido é analisado e o usuário responsável verifica que o incidente e/ou erro relatado foi solucionado de maneira satisfatória de acordo com o padrão de qualidade definido para o software;
- FECHADO: ao fim do projeto, os relatórios são fechados, isso significa que o ciclo de vida do incidente e/ou erro foi avaliado e que esse foi solucionado adequadamente sendo, portanto, encerrado definitivamente.

Os elementos de modelagem de representação do fluxo de *status* são descritos na Tabela 4.1.

Tabela 4.1 - Elementos de Modelagem do Fluxo de Status (Elaborado pelo Autor).

SÍMBOLOS	SIGNIFICADO
	Representa o início ou o fim do fluxo de *status* do Mantis.
	Representa um *status* a ser executada por um usuário.
	Representa uma resolução, ou seja, a condição em que o *status* está.
	Representa um papel a ser exercido por um usuário do sistema.
	Representa a transição entre os *status* do fluxo.

Papéis são as classificações do usuário de acordo com suas funções e responsabilidades nos processos de desenvolvimento e de teste de software, sua

participação na execução das etapas e atividades. Esses usuários possuem permissão, no Mantis, para tratar o incidente e/ou erro de acordo com o papel que exerce no processo de desenvolvimento e de teste do software.

Foram definidos dois tipos de fluxos a serem realizados no Mantis, a saber:

- Fluxo de *status* de requisitos, usado nas atividades de revisão da documentação dos processos de desenvolvimento e de teste de software. Este fluxo é utilizado para o relato e o registro de incidentes detectados. A Figura 4.2 ilustra o fluxo de status de requisitos adotado no processo SwTest;

- Fluxo de *status* de teste, usado nas atividades de execução dos testes do processo SwTest. Este fluxo é utilizado para o relato e o registro de erros detectados. A Figura 4.3 ilustra o fluxo de status de teste adotado no processo SwTest.

Foi criado um manual de uso e configuração padrão do sistema para a empresa. Houve a realização de treinamentos para os funcionários a fim de proporcionar o aprendizado da utilização e da importância dos novos fluxos anteriormente apresentados.

Ainda, para que os funcionários tivessem fácil acesso, os fluxos de *status* de requisito e de testes definidos para uso na empresa foram disponibilizados na página *web* da empresa.

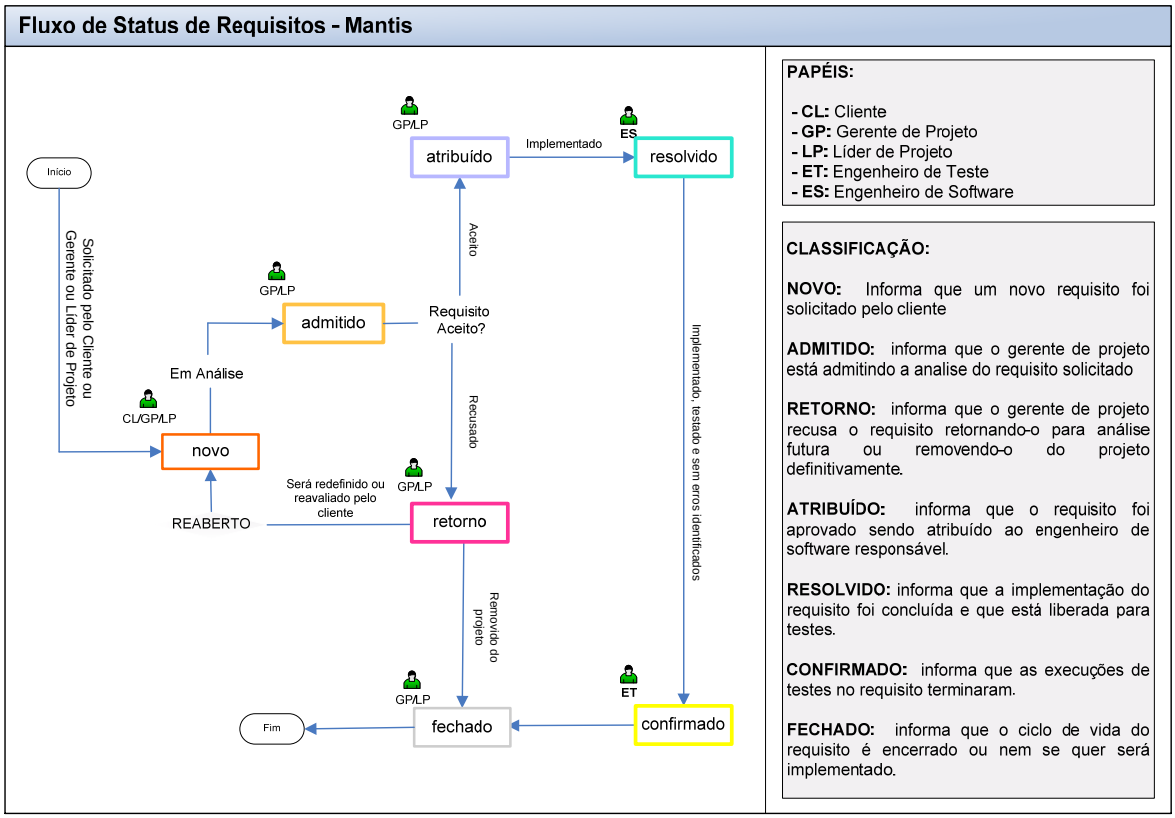


Figura 4.2 - Fluxo de Status de Requisitos no Mantis (Fonte: A Empresa).

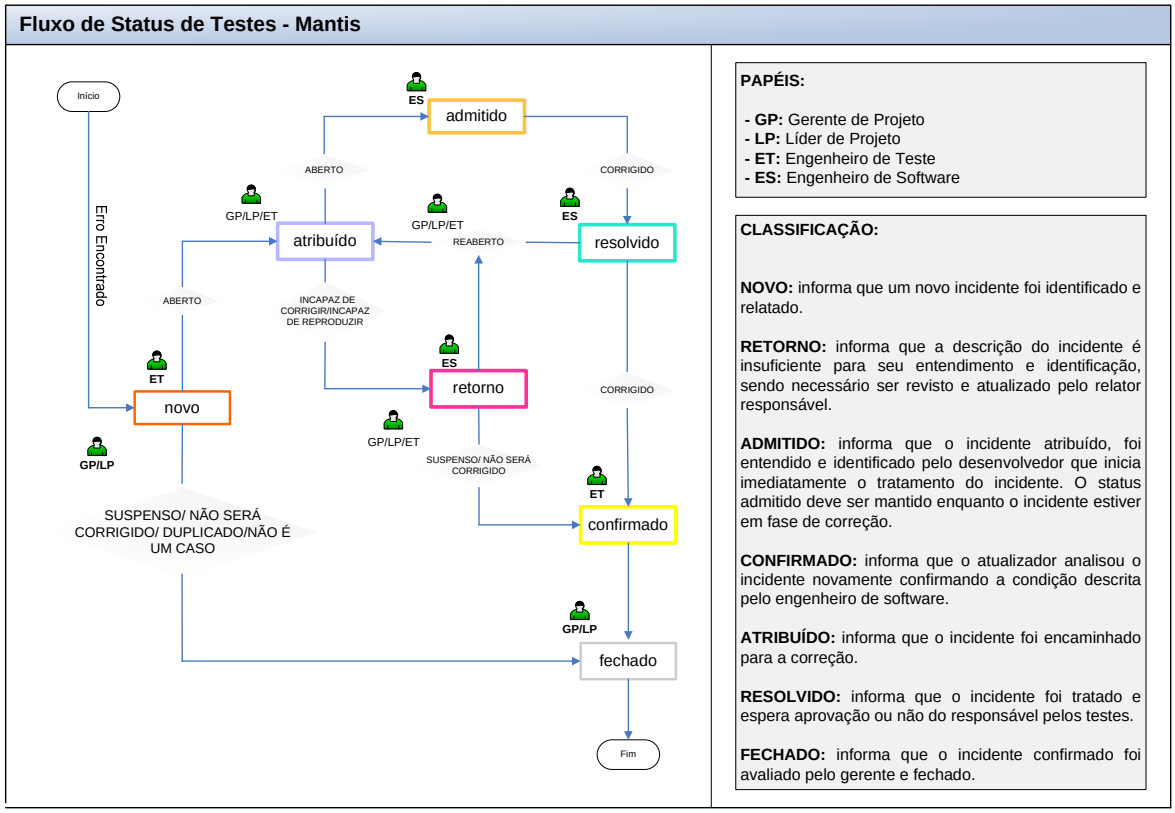


Figura 4.3 - Fluxo de Status de Testes no Mantis (Fonte: A Empresa)

4.3 Etapas do Processo SwTest

O processo SwTest envolve quatro etapas que devem ser executadas durante o próprio processo de desenvolvimento de software: planejamento, projeto, execução (e coleta dos resultados) e avaliação dos resultados coletados.

A Figura 4.4 apresenta o fluxo das etapas adotadas no processo SwTest, sendo que suas etapas devem ser executadas paralelamente às etapas do processo de desenvolvimento de software da empresa e são apresentadas adiante.

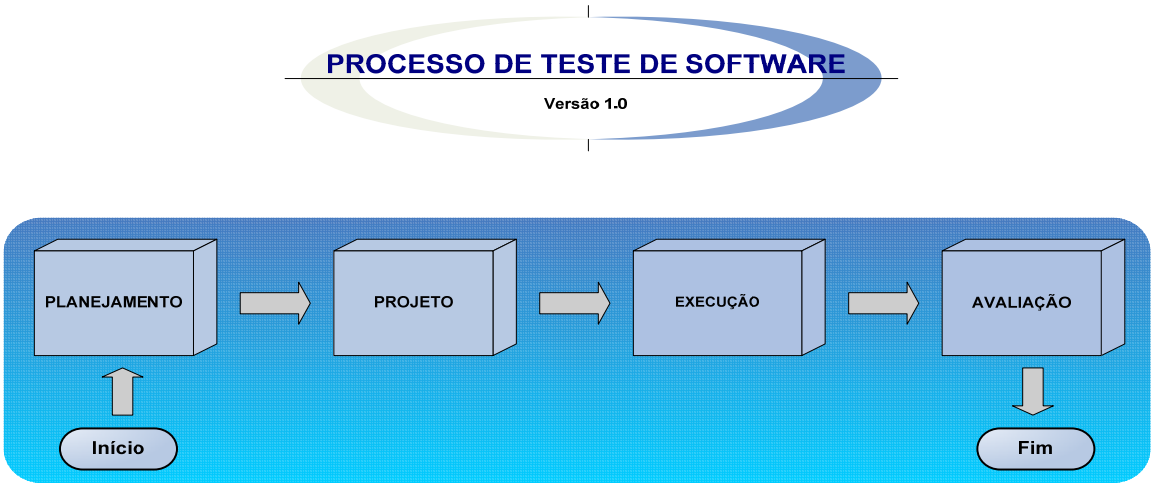


Figura 4.4 - Processo de Teste de Software (Fonte: A Empresa).

O processo SwTest é baseado em artefatos de testes declarados na norma IEEE 829 e têm suas atividades definidas e ordenadas de forma a permitir que o teste seja uma das atividades de garantia da qualidade do software desenvolvido.

Para cada etapa do processo SwTest, foram definidas as atividades necessárias para que fossem gerados os produtos de trabalho que, nesse contexto, são os artefatos que devem ser produzidos ao longo do processo de desenvolvimento do software. Para cada artefato de entrada, geram-se artefatos de saída.

Uma atividade possui um propósito e artefatos de entrada e de saída. As atividades são descritas com o objetivo de atingir o propósito de processo, ou seja, são executadas a fim de alcançar o objetivo descrito no processo. Os artefatos de entrada podem ser *templates* de documentos, documentos preenchidos, padrões do processo ou elementos informativos. Os artefatos de saída podem ser um ou mais artefatos de entrada atualizados

ou preenchidos ou ainda um novo artefato de trabalho produzido de acordo com a necessidade surgida durante o processo de desenvolvimento ou de teste do software.

Para auxiliar as informações do fluxo do processo, foi criado um documento de referência para as atividades das fases. Neste documento, para cada atividade, são apresentados:

- Finalidade - objetivos, elementos e/ou condições necessárias para iniciar a execução das atividades;
- Artefatos de Entrada e Saída - dados e recursos necessários para a execução das atividades e gerados pela execução das atividades;
- Papel - responsável pela execução da atividade;
- Ferramentas utilizadas.

A Figura 4.5 apresenta como as atividades de uma etapa do processo SwTest são documentadas.

Primeira Etapa: Planejamento

Finalidade

- Entender os objetivos do software desenvolvido e planejar as atividades que serão realizadas.
- Definir os critérios para o início do processo de teste, determinando elementose/ou condições necessários para iniciar a execução das tarefas.

Artefatos de Entrada:	**Artefatos de Saída:**
• Template do Plano de Teste-xPLT • Documento de Requisitos-DRE • Plano de Projeto-PLP • Documento de Caso de Uso - DCU • Fluxograma dos Testes - FLT • Cronograma do Projeto - CRO	• Plano de Teste - PLT

Papel: Gerente de Projeto e Engenheiro de Teste

Ferramentas: Word e Ferramenta de Gerência de Configuração

Passos:

1. Analisar o DCU, juntamente com o DRE, definindo quais testes específicos devem ser realizados;
2. Definir as atividades de planejamento dos testes relevantes ao projeto;
3. Preencher no PLT as informações necessárias para a realização das atividades de teste de maneira organizada e coerente com as etapas de desenvolvimento do projeto;
4. Armazenar o PLT na ferramenta de gerência de configuração utilizado no respectivo projeto;
5. Comunicar os resultados do planejamento ao gerente de projeto e apresentar às equipes de teste e de desenvolvimento.

Figura 4.5 - Documentação de atividades de uma das etapas do processo SwTest (Fonte: A Empresa).

Para cada atividade, é atribuído um ou mais responsáveis para a sua execução. As próximas subseções descrevem as quatro etapas do processo, as finalidades, as atividades, os artefatos de entrada e de saída e os papéis.

4.3.1 Planejamento

Na etapa de planejamento, em conjunto com a área técnica responsável pelo desenvolvimento do software, deve-se definir o que vai ser testado (escopo dos testes), quem vai testar, quando serão realizados os testes e os recursos necessários para a realização das atividades de testes do software. Esta etapa prevê a elaboração de um documento chamado de Plano de Testes de software, que terá como insumos alguns artefatos de especificação de requisitos, para atender a finalidade do software a ser testado.

A etapa de planejamento do processo SwTest deve ser realizada paralelamente a etapa de planejamento do processo de desenvolvimento do software.

A seguir, são apresentados a finalidade, os artefatos, os papéis e as atividades do processo SwTest correspondente a etapa de planejamento:

1. Finalidade

Na etapa de planejamento, o engenheiro de teste entende os objetivos do software a ser testado e define os critérios para o início do processo SwTest, determinando elementos e/ou condições necessários para iniciar a execução das tarefas.

2. Artefatos

A seguir, são apresentados os artefatos de entrada e de saída utilizados e gerados na execução da etapa de planejamento do processo SwTest:

- **Entrada:** Documento de Requisitos, Plano de Projeto, Documento de Caso de Uso e Cronograma do Projeto;
- **Saída:** Plano de Teste que descreve o planejamento das atividades acima citadas.

A definição do Plano de Teste é conduzida no início do projeto, antes que outros detalhamentos do fluxo de trabalho comecem. O Plano de Teste é apresentado no Apêndice A.

3. Papel

Na etapa de planejamento do processo SwTest, os participantes na execução das atividades são:

- **Gerente de Projeto:** responsável por determinar, junto com o engenheiro de teste, o escopo dos testes, identificar o que será e o que não será testado e analisar o Plano de Teste;
- **Engenheiro de Teste de Software:** responsável por definir o escopo dos testes, especificar os artefatos, padrões e ferramentas, identificar os itens que serão e os que não serão testados e elaborar o Plano de Teste.

4. Atividades

As ações necessárias para produzir as saídas da etapa de planejamento do processo SwTest são:

- Definir o Escopo dos Testes;
- Determinar os artefatos, Padrões, Ferramentas e Ambiente de Testes;
- Identificar itens que serão ou não testados;
- Estabelecer abordagens e estratégias de teste que serão aplicadas na execução dos testes, elaborando um Fluxograma dessas estratégias;
- Definir ambiente de testes (software a ser testado e as ferramentas de apoio);
- Elaborar o cronograma das atividades;
- Estabelecer responsabilidades dentro da equipe definida;
- Estabelecer treinamentos necessários;
- Identificar riscos;
- Definir o critério de aprovação dos testes;

- Elaborar o Plano de Teste que será aplicado ao software em desenvolvimento.

A Figura 4.6 ilustra as atividades e a organização da etapa de planejamento do processo SwTest.

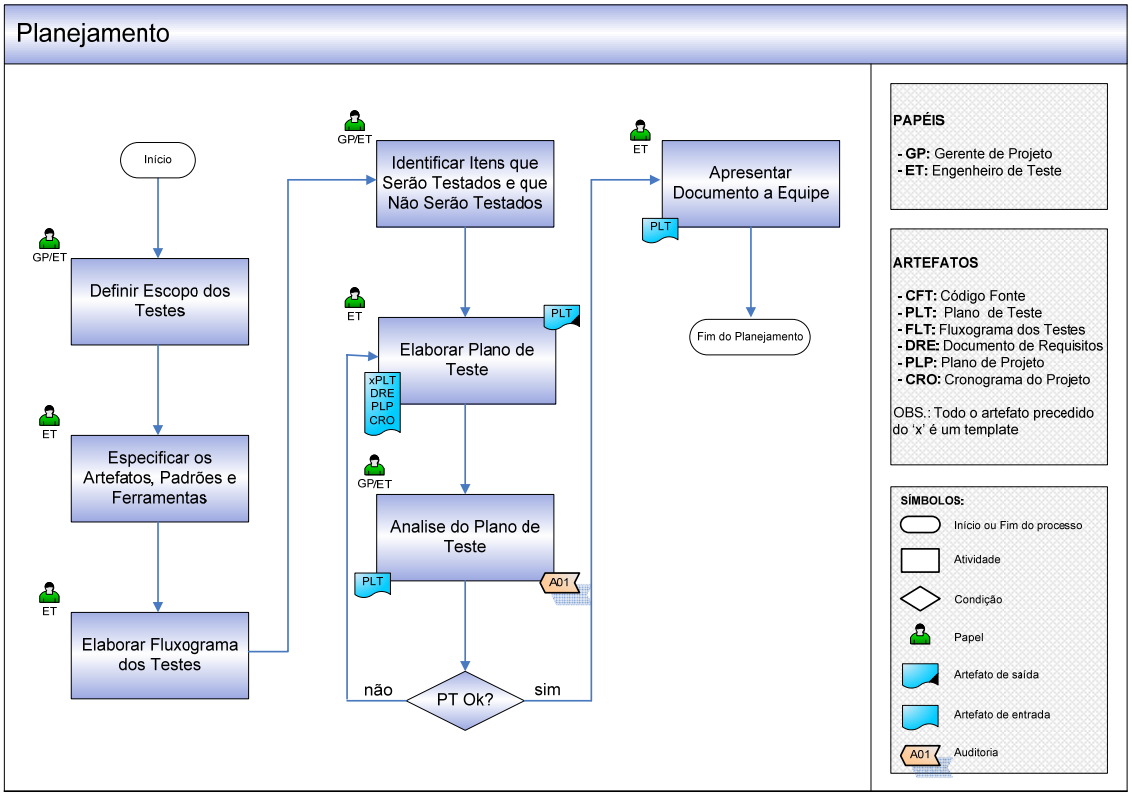


Figura 4.6 - Etapa de Planejamento do Processo SwTest. (Fonte: A Empresa).

4.3.2 Projeto

Na etapa de projeto, a bateria de testes deve ser definida. Para tal, os casos de teste e os procedimentos de teste são definidos, juntamente com a ordem de sua execução.

Os casos de teste contêm dados de entrada e valores de saída esperados para cada instância de teste e as pré-condições necessárias para que o caso de teste possa ser executado. Os valores de entrada são escolhidos de acordo com critérios que maximizam a cobertura dos testes.

Os procedimentos de teste contem uma seqüência de passos a serem executados para a realização de um conjunto de testes semelhantes. Cada procedimento de teste corresponde a um roteiro para: instalação da aplicação a ser testada, instalação de

ferramentas de apoio e realização de um caso de uso (teste funcional) e de *scripts* de teste (no caso de utilização de ferramentas de automação de testes).

Os casos de teste e os procedimentos de teste são documentados no Documento de Caso de Teste específico para cada requisito do projeto em desenvolvimento.

A seguir, são apresentados a finalidade, as atividades, os artefatos e os papéis dos envolvidos no fluxo do processo SwTest correspondente a etapa de Projeto:

1. Finalidades

Na etapa de projeto, o engenheiro de teste define a bateria de teste, estabelecendo os procedimentos de teste, os casos de teste e a sua ordem de execução. Define-se "como fazer".

2. Artefatos

A seguir, são apresentados os artefatos de entrada e de saída utilizados e gerados na execução da etapa de projeto do processo SwTest:

- **Entrada:** Plano de Teste de Software; Manuais de instalação e de operação do software e das ferramentas de apoio; Matriz de Ratreabilidade, Documento de Requisito e Documentos de Caso de Uso;

- **Saída:** Relatórios de Revisão dos Documentos do Projeto; Documentos de Caso de Teste e Plano de Teste de Integração.

3. Papéis

Na etapa de projeto do processo SwTest, os participantes na execução das atividades são:

- **Engenheiro de Teste:** responsável por elaborar os Documentos de Caso de Teste e o Plano de Teste de Integração. Também é responsável por executar as revisões nos documentos do projeto em desenvolvimento e preparar o ambiente de teste;

- **Engenheiro de Software:** responsável por executar correções nos documentos revisados de acordo com o relatado no sistema Mantis;

- **Gerente de Projeto:** responsável por analisar e atribuir correções dos documentos do projeto revisados pelo engenheiro de teste, além de avaliar o Plano de Teste de Integração.

4. Atividades

As ações necessárias para produzir as saídas da etapa de projeto do processo SwTest, são:

- Executar Revisões nos Documentos relacionados ao desenvolvimento;
- Relatar Falhas e Inconsistências Encontradas nesses documentos;
- Elaborar Documentos de Casos de Teste, baseados nos Documentos de Casos de Uso, nos casos de teste definidos e nos procedimentos de teste;
- Analisar a matriz de rastreabilidade e elaborar o Plano de Teste de Integração;
- Analisar o Plano de Integração;
- Planejar a Execução dos testes, baseada nas estratégias definidas no Plano de Teste;
- Preparar Ambiente de Teste (instalar hadware, software, etc);
- Apresentar Documentos Elaborados à equipe e, se necessário, realizar treinamentos.

A Figura 4.7 ilustra a etapa de projeto do processo SwTest e as atividades, os artefatos e os responsáveis pela execução dessas atividades.

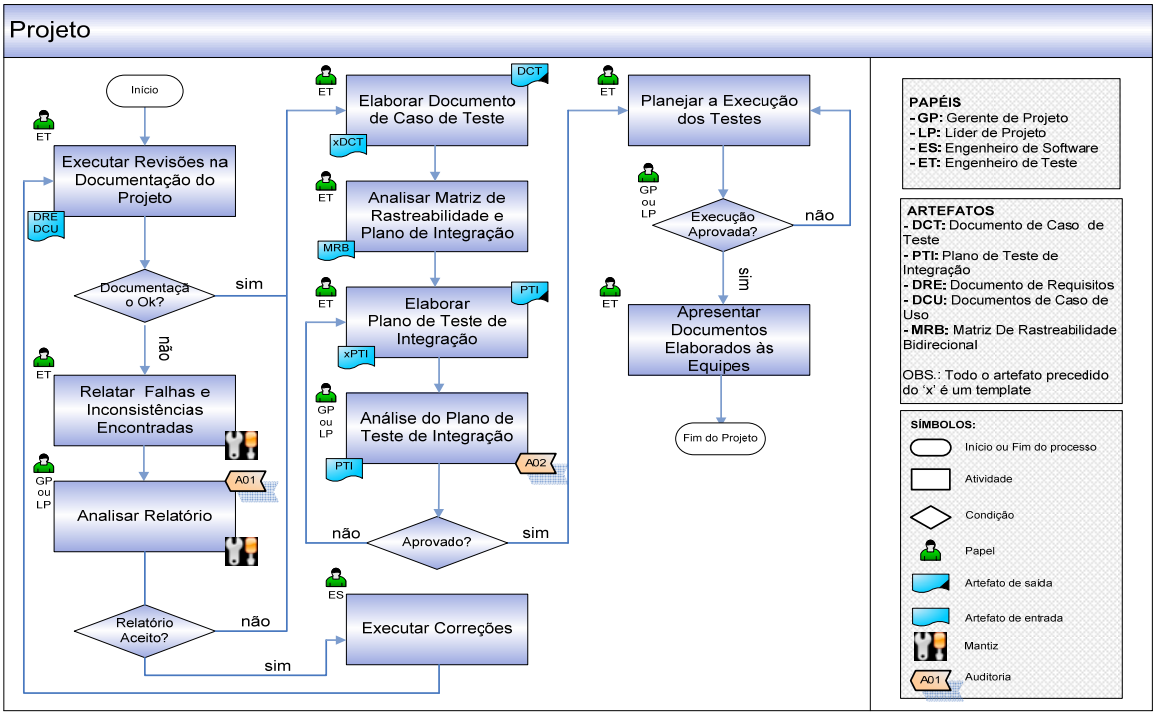


Figura 4.7 - Etapa de Projeto do Processo SwTest (Fonte: A Empresa).

4.3.3 Execução

A etapa de execução dos testes foi baseada nas instruções de testes da norma ISO/IEC 12119, que recomenda como um produto deve ser testado em relação aos requisitos de qualidade.

Nesta etapa do processo SwTest, o objetivo é realizar os testes definidos nas etapas anteriores, a fim de garantir que a implementação de cada requisito do software seja testada em conformidade com o requisito e que o software esteja pronto para ser liberado para implantação final.

Os testes planejados são executados e os resultados são registrados no Documento de Caso de Teste (Apêndice B). Os erros detectados são descritos no relatório gerado no Mantis, como pode ser visto na Figura 4.8 .

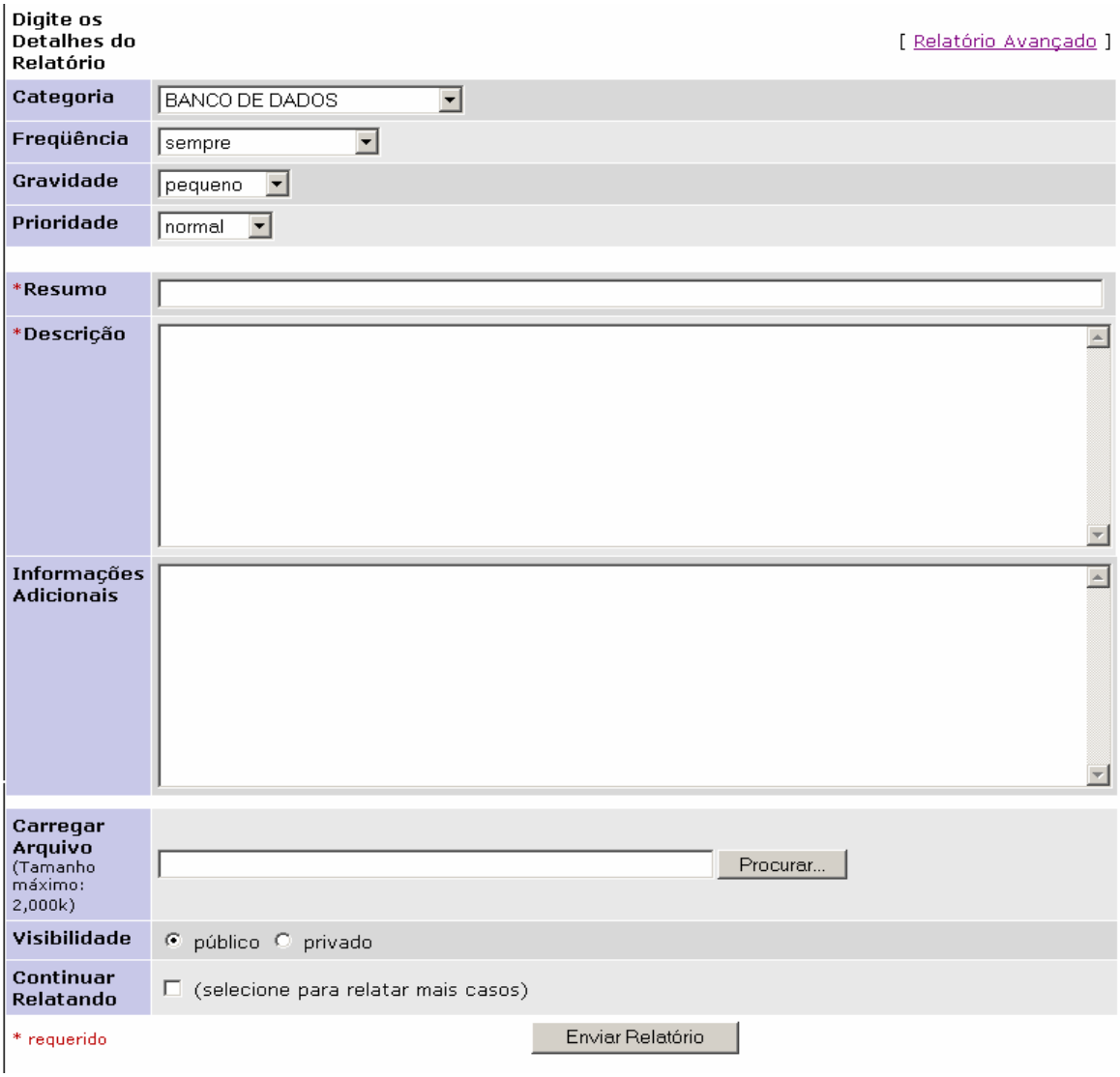


Figura 4.8 - Página de Relatório do Mantis (Fonte: A Empresa).

As atividades da etapa de execução dos testes são realizadas de maneira sistemática e organizada, fornecendo dados e informações relevantes ao tratamento dos erros detectados. Sendo assim, os testes feitos de maneira sistemática são um bom ponto de partida para medições para a avaliação das atividades de teste e do processo SwTest que será apresentado na próxima subseção.

A seguir são apresentados a finalidade, as atividades, os artefatos e os papéis dos envolvidos no fluxo do processo SwTest correspondente a etapa de execução:

1. Finalidades

Na etapa de execução dos testes, o engenheiro de teste realiza a avaliação contínua dos requisitos testados, registrando as informações necessárias para diagnosticar e resolver

os problemas identificados. O objetivo é fornecer *feedback* nas áreas de risco potencial para a qualidade.

2. Artefatos

A seguir, são apresentados os artefatos de entrada e de saída utilizados e gerados na execução da etapa de execução do processo SwTest:

- **Entrada:** Documentos de Caso de Teste, Plano de Teste de Integração;
- **Saída:** Relatório de Erros. Para esse relatório, deve ser usado o sistema Mantis.

3. Papéis

Na etapa de execução do processo SwTest, os participantes na realização das atividades são:

- **Engenheiro de Teste:** responsável por executar os testes anteriormente apresentados, garantir o uso e preenchimentos dos artefatos de entrada e saída desta etapa e gerar os relatórios de erros;
- **Engenheiro de Software:** responsável por corrigir os erros detectados;

4. Atividades

As ações necessárias para produzir as saídas da etapa de execução dos testes do processo SwTest são:

- Executar os testes de unidade de acordo com os Documentos de Caso de Teste, verificando: 1) Interface com o módulo, para ter a garantia das informações fluírem para dentro e para fora da unidade que se encontra em teste; 2) A estrutura de dados locais, para ter a garantia de que os dados armazenados temporariamente mantêm sua integridade durante os passos de execução de um algoritmo; 3) As condições de limite, para ter a garantia de que a unidade opera adequadamente nos limites estabelecidos para demarcarem ou restringirem o processamento; 4) Caminhos independentes através da estrutura de controle, para ter a garantia de que as instruções de uma unidade foram executadas pelo menos uma vez; e 5) Se os caminhos de tratamento de erros são testados;

- Executar testes de integração de acordo com o estabelecido no Plano de Teste de Integração, verificando se os módulos de nível mais baixo são combinados em construções que executam uma subfunção específica do software;
- Executar testes de sistema para verificar e validar o sistema ou parte dele;
- Registrar os testes e relatar os erros encontrados durante as diversas etapas de execução;
- Garantir a qualidade do produto de software desenvolvido.

A Figura 4.9 ilustra a etapa de execução do processo SwTest e as atividades, os artefatos e os responsáveis pela realização dessas atividades.

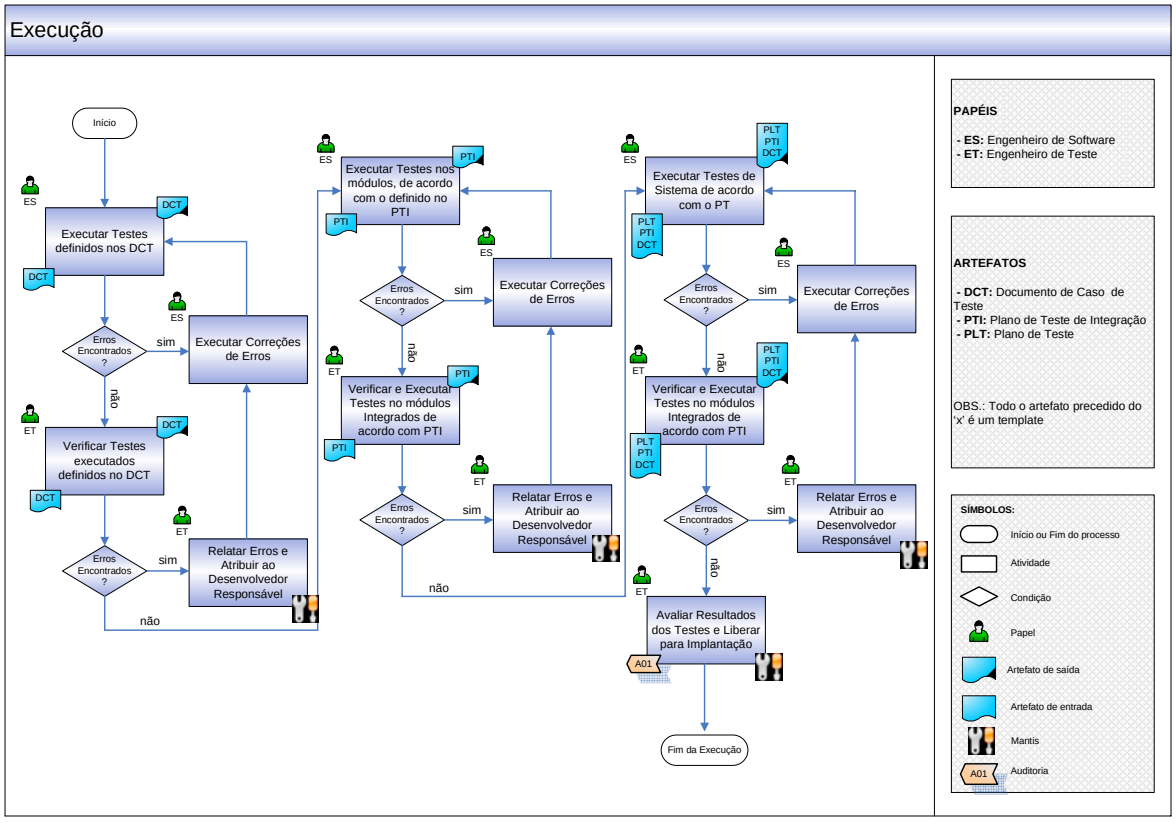


Figura 4.9 - Etapa de Execução do Processo SwTest (Fonte: A Empresa).

4.3.4Avaliação

Nesta etapa, é realizado o teste de aceitação do software e iniciam-se as atividades de avaliação das etapas anteriores.

A partir da etapa de avaliação, evidências dos resultados obtidos durante as etapas anteriores são analisadas e é verificado se o software desenvolvido atende as especificações feitas pelo cliente e demonstra a conformidade com os requisitos.

É realizada uma avaliação das equipes de teste e de desenvolvimento e também do gerente do projeto.

O objetivo da empresa é, a partir dos dados gerados pelo processo SwTest corrigir desvios e modificar conseqüentemente as prioridades e as ações corretivas a serem tomadas; ou seja, aprimorar os processos de desenvolvimento e de teste de software da empresa com o aprendizado.

A seguir, são apresentados a finalidade, os artefatos, os papéis e as atividades do processo SwTest correspondente a etapa de avaliação:

1. Finalidades

O objetivo desta etapa é, a partir dos resultados dos testes, verificar se as condições de completeza e de sucesso dos testes definidos no Plano de Testes são satisfeitas, avaliar os resultados dos testes e elaborar um relatório de desenvolvimento do software.

2. Artefatos

A seguir, são apresentados os artefatos de entrada e de saída utilizados e gerados na execução da etapa de avaliação do processo SwTest:

- **Entrada**: Plano de Teste de Software, Documento de Caso de Teste, Plano de Integração;
- **Saída**: Planilha de Avaliação dos Testes, Relatório de Desempenho do Projeto, Plano de Melhorias.

3. Papéis

Na etapa de avaliação do processo SwTest, os participantes na execução das atividades são:

- **Engenheiro de Teste:** responsável por analisar os erros detectados durante o teste de aceitação realizado pelo cliente/usuário, e atribuir, caso concorde com o relatório do cliente, os erros detectados aos responsáveis pela

correção. Ainda é de sua responsabilidade avaliar os resultados dos testes e propor melhorias para futuros projetos;

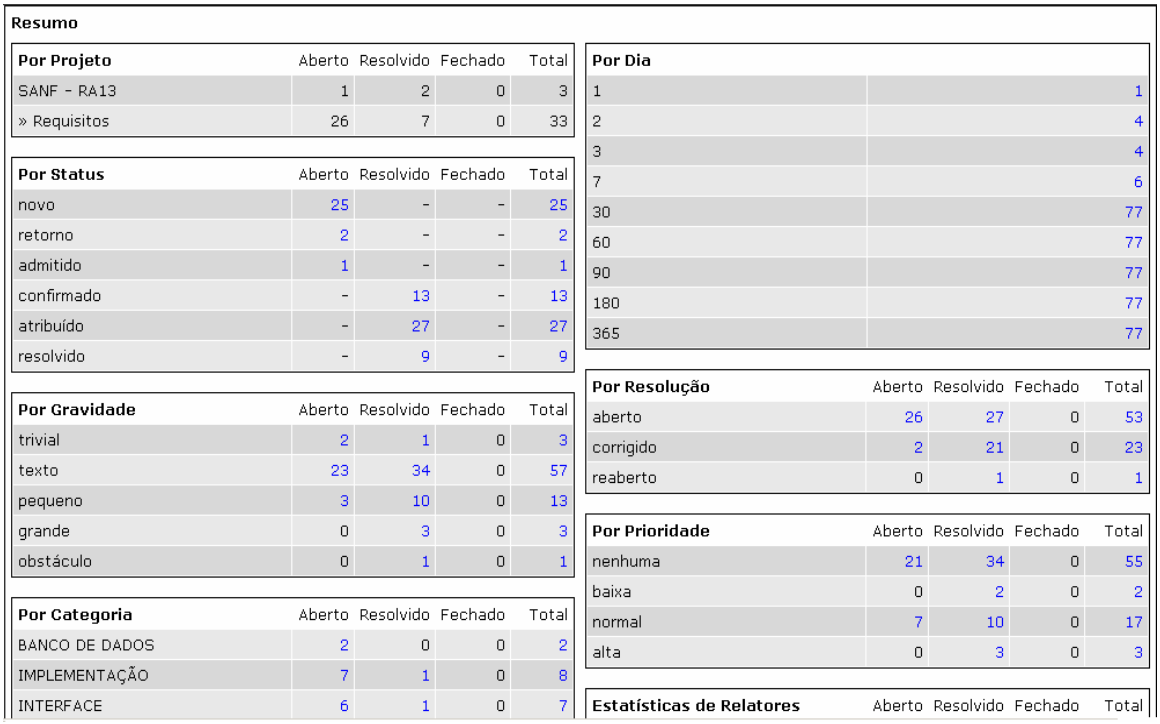
- **Engenheiro de Software:** responsável por corrigir os erros detectados durante o teste de aceitação do sistema.

4. Atividades

As ações necessárias para produzir as saídas da etapa de avaliação dos testes do processo SwTest são:

- Verificar se o produto está realmente pronto, de acordo com o planejado;
- Verificar a autenticidade dos erros detectados durante o teste de aceitação do software realizado pelo cliente/usuário, relatados no sistema Mantis;
- Corrigir erros encontrados durante a realização dos testes de aceitação, realizados pelo cliente/usuário;
- Avaliar os resultados dos testes registrados nos Documentos de Caso de Teste, Plano de Integração e no Sistema Mantis. A Figura 4.10 ilustra a página de resumo estatístico gerado pelo sistema Mantis;
- Gerar uma Planilha de Avaliação dos Testes, destacando “pontos de sucesso” e “pontos de fracasso”, sugerindo soluções para os problemas e registrar o desempenho dos atores do projeto;
- Realizar reunião de fechamento do projeto e apresentar os resultados a fim de proporcionar abertura para “idéias” para melhorar a organização e o processo.

A Figura 4.11 ilustra a etapa de avaliação do processo SwTest e as atividades, os artefatos e os responsáveis pela realização dessas atividades.



Resumo

Por Projeto	Aberto	Resolvido	Fechado	Total
SANF - RA13	1	2	0	3
» Requisitos	26	7	0	33

Por Status	Aberto	Resolvido	Fechado	Total
novo	25	-	-	25
retorno	2	-	-	2
admitido	1	-	-	1
confirmado	-	13	-	13
atribuído	-	27	-	27
resolvido	-	9	-	9

Por Gravidade	Aberto	Resolvido	Fechado	Total
trivial	2	1	0	3
texto	23	34	0	57
pequeno	3	10	0	13
grande	0	3	0	3
obstáculo	0	1	0	1

Por Categoria	Aberto	Resolvido	Fechado	Total
BANCO DE DADOS	2	0	0	2
IMPLEMENTAÇÃO	7	1	0	8
INTERFACE	6	1	0	7

Por Dia	
1	1
2	4
3	4
7	6
30	77
60	77
90	77
180	77
365	77

Por Resolução	Aberto	Resolvido	Fechado	Total
aberto	26	27	0	53
corrigido	2	21	0	23
reaberto	0	1	0	1

Por Prioridade	Aberto	Resolvido	Fechado	Total
nenhuma	21	34	0	55
baixa	0	2	0	2
normal	7	10	0	17
alta	0	3	0	3

Estatísticas de Relatores	Aberto	Resolvido	Fechado	Total

Figura 4.10 - Página de Visualização dos Resumos Estatísticos do Mantis.

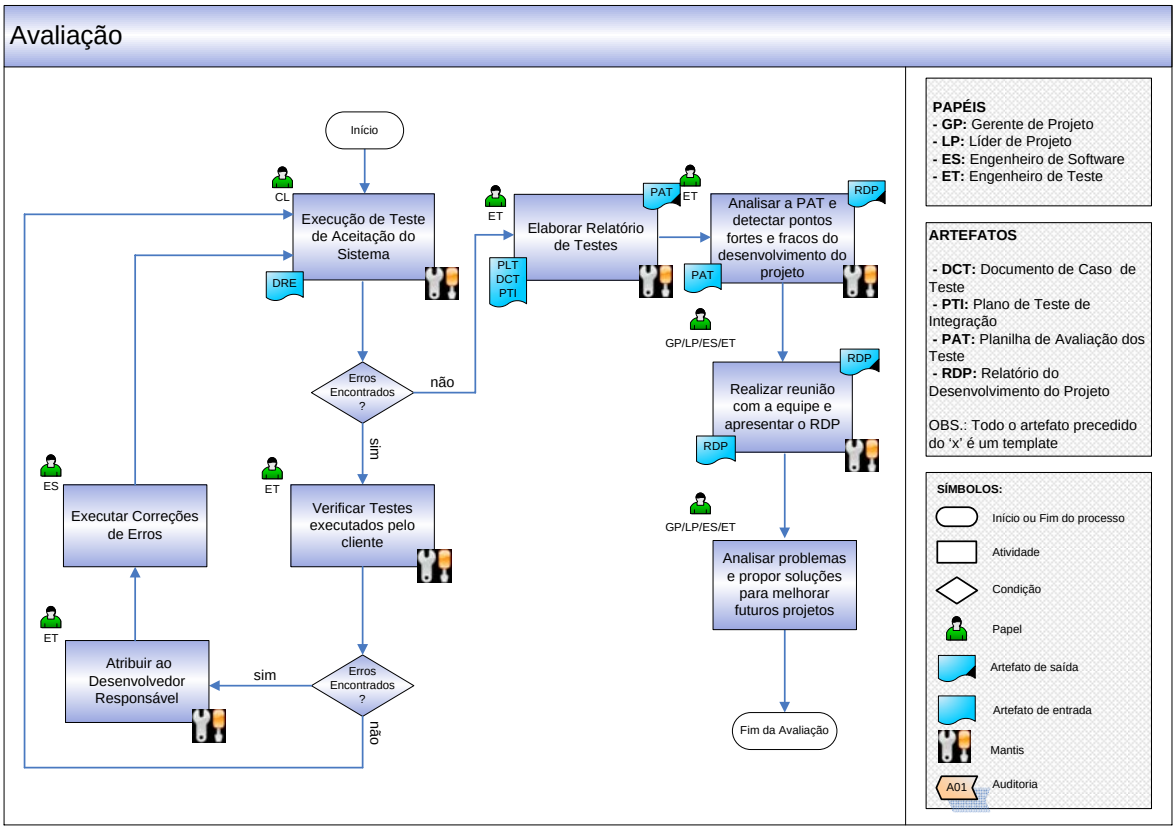


Figura 4.11 - Etapa de Avaliação do Processo SwTest (Fonte: A Empresa).

4.4 Considerações Finais

O processo SwTest propõe-se a melhorar a comunicação entre o engenheiro de software e o engenheiro de testes e, também, a definição da documentação necessária para a boa prática dos testes, definindo as fases em que serão aplicados e de que maneira serão avaliados. Desta forma, o responsável pelo projeto passa a ter mais controle das atividades de suporte.

Mesmo não sendo um projeto com características inovadoras, é importante salientar que, no mercado onde a Empresa está inserida, fábrica de software e consultoria, é necessário que se tenha a consciência da importância de ter um processo de testes bem definido, a fim de garantir a qualidade do software desenvolvido.

A experiência adquirida com a realização deste projeto e o uso do processo e da metodologia possibilitam à Empresa, compartilhar e trocar experiências com as demais empresas de desenvolvimento de software do mercado.

Para a institucionalização do processo SwTest na empresa foram realizados treinamentos com todos os colaboradores envolvidos, onde foi exposto como o processo SwTest deve ser utilizado, a forma de realizar as atividades e o uso correto dos *templates* criados. O treinamento consistiu na conscientização da realização de testes de software, apresentando os conceitos envolvidos, e do uso de um processo de teste, no caso SwTest.

Para facilitar o acesso dos funcionários ao processo SwTest foi disponibilizado na *intranet* da empresa o fluxograma das etapas, das atividades, documentos e ferramentas utilizadas no processo SwTest.

Foi observada também a necessidade de criar *templates* que tratam os procedimentos de teste e uma planilha de avaliação dos testes realizados.

5 CONSIDERAÇÕES FINAIS

5.1 Conclusão

Neste trabalho, procurou-se ressaltar a importância da garantia e do controle da qualidade em produtos de software, na conquista da satisfação do usuário. Enfatizou-se que, para garantir a qualidade do processo e do software desenvolvido, é necessário ter um processo de teste de software bem definido, pois a qualidade do software está intimamente relacionada às atividades de teste de software nele aplicado.

O problema levantado está baseado em dois fatos principais:

- Em geral, a atividade de teste é feita de forma pouco sistemática e quase nunca documentada;
- Mesmo com a ajuda de ferramentas que automatizam parte da atividade de teste, inúmeras vezes a execução dos casos de teste e sua posterior análise são atividades custosas ou às vezes inviáveis.

Entende-se que a preocupação pelo teste de software deve estar presente desde o início do desenvolvimento, na elaboração das atividades de desenvolvimento de software, de forma que as atividades relacionadas com o teste possam ser amadurecidas de forma adequada.

A definição do processo SwTest e da metodologia de testes para a empresa de pequeno porte desenvolvedora de software foi um passo fundamental no esforço de melhoria contínua de processos formais a serem executados durante as etapas de desenvolvimento do software, como forma de aumentar a qualidade do software desenvolvido e minimizar o custo deste desenvolvimento.

Conclui-se também que, para testar um software, é necessário um bom treinamento, ter experiência, conhecer bem os software e ter criatividade. Mas, é necessário saber que, apesar de tudo isso, por mais tempo e energia gastos nos testes, não se terá um software livre de erros.

5.2 Contribuições

Os seguintes pontos se destacam como contribuições do presente trabalho:

- O detalhamento dos tipos de testes necessários para desenvolvimento do software, possibilitando a definição de testes mais completos, auxiliando na garantia de qualidade do software desenvolvido, uma vez que as suas características específicas serão contempladas durante as atividades relacionadas com o teste;

- O desenvolvimento sistemático dos testes de software facilita o seu entendimento, possibilitando a construção de testes mais completos e corretos. Além disso, facilita a elaboração dos testes, visto que é possível determinar objetivos específicos para serem alcançados com os testes;

- A realização sistemática das atividades relacionadas com teste desde o início do processo de desenvolvimento contribui para que os erros sejam identificados logo no início e, conseqüentemente, diminui os custos necessários para a sua correção;

- O detalhamento das informações adicionadas ao processo de teste amadurece o entendimento do software, contribuindo para a obtenção de software de qualidade;

- A elaboração dos *templates* de Plano de Teste, do Documento de Caso de Teste e do Plano de Teste de Integração feita, a partir de informações identificadas nos modelos de processo de desenvolvimento de software, contribui para a redução dos esforços necessários para a execução das atividades de teste;

- A estrutura definida para o desenvolvimento dos testes foi centralizada. Mesmo considerando um custo maior com recursos humanos voltados 100% para os testes; atualmente, pode-se garantir que as atividades de teste serão realizadas de forma disciplinada e coordenada e não sendo negligenciadas pelas equipes de desenvolvimento. Anteriormente, cada gerente de projeto, baseado em suas crenças, dava ou não ênfase a estas atividades e o reflexo era imediato à implantação dos sistemas (quantidade de erros após a implantação). Com o uso do processo SwTest, os gerentes de projeto atuantes no setor de fábrica de software da empresa serão providos de métricas que mostram os ganhos da utilização do processo;

- O planejamento e a execução dos testes realizados durante as fases do desenvolvimento do software foi uma das mudanças mais expressivas. Nos setores onde havia um processo de testes instituído, ele ocorria após a construção do software e os erros de análise e de projeto não eram antecipados à construção. Logo, a definição das etapas do processo SwTest possibilitou visibilidade da qualidade do software que é construído, desde o início do desenvolvimento, e está antecipando erros e problemas;

- O processo e a metodologia de testes passaram a gerar subsídios para a equipe de desenvolvimento, possibilitando fazer melhorias na gestão do processo de desenvolvimento;

- No final de cada projeto, são relacionadas as lições aprendidas durante o desenvolvimento de cada sistema. Com base nestas informações, que são divulgadas aos demais gerentes de projeto, podem-se fazer melhorias no desenvolvimento dos próximos sistemas.

A partir da implantação da metodologia e do uso do processo SwTest, são coletadas métricas que reforçam a sua importância para a geração de produtos com mais qualidade.

5.3 Trabalhos Futuros

Alguns trabalhos futuros que podem ser desenvolvidos a partir deste são:

- Avaliar o processo em projetos reais;

- A partir dessa avaliação, propor melhorias para o processo de desenvolvimento de software da empresa e do próprio SwTest;

- Automatizar o processo através da adesão ao uso de ferramentas de teste de software;

- Elaborar os *templates* para Procedimentos de Teste, para Planilha de Avaliação dos Testes e para o Plano de Melhorias;

- Propor métricas de avaliação para o SwTest.

Referencial Bibliográfico

ANDRADE, Ana Luísa P.; Oliveira, Angelina P. de; Capovilla, Celso R.; Rego, Claudete M.; Souza, Eduardo P. de; Martinex, Márcia Regina M; Aguayo, Maria Tereza V.; Jino, Mario. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo.2004.** Ampla Consultoria em Informação, Campinas – SP. Centro de Pesquisas Renato Archer (CenPRA),

Aplicação da Norma ISO/IEC 12119 na Avaliação da Qualidade de Produtos de Software. VII Conferência Internacional de Tecnologia 80 de Software: Qualidade de Software - VII CITS – Curitiba – Junho de 1996, págs: 75 – 89.

Crespo, A. N., Silva, O. J., Borges, C. A., Salviano, C. F., Argollo, M., Jino, M. (2004) **"Uma metodologia para teste de Software no Contexto da Melhoria de Processo",** In: III Simpósio Brasileiro de Qualidade de Software (SBQS 2004), Brasília.

FERREIRA, A. B. H. **Novo dicionário Aurélio da língua Portuguesa.** Ed. Nova Fronteira, 1986.

The Institute of Electrical and Electronics Engineers. IEEE Std 829: Standard for Software Test Documentation. New York: IEEE Computer Society, September, 1998.

ISO/IEC 12119 International Standard. Information Technology-Software Packages – Quality requirements and testing; October 1994.

ISO/IEC 9126, International Standard. Information Technology – Software Product Evaluation – Quality characteristics and guidelines for their use; 1991.

ISO/IEC 9126-1, International Standard. Information Technology – Software Quality Characteristics and Metrics – Part 1: Quality Characteristcs and Sub-Characteristics; June, 1995 (Working Draft).

ISO/IEC 9126-2, International Standard. Information Technology – Software Quality Characteristics and Metrics – Part 2: External Metrics; November, 1995 (Working Draft).

ISO/IEC 9126-1, International Standard. Information Technology – Software Quality Characteristics and Metrics – Part 3: Internal Metrics; November, 1995 (Working Draft).

JONES, C., **Programming Productivity**, McGraw – Hill, 1986.

JUNG, C. **Metodologia Para Pesquisa & Desenvolvimento –Aplicada a Novas Tecnologias, Produtos e Processos**. Rio de Janeiro: Axcel Books, 2004. 162 p.

MALDONADO, J.C. et al. **Introdução ao teste de software.** In: X IV S IMPÓSIO 90 BRASILEIRO DE ENGENHARIA DE SOFTWARE, João Pessoa, PB, outubro de 2000.

MYERS, Glenford J. **The Art of Software Testing**; Revised and Update by Tom Badgett and Tood Thomas whit Corey Sander – 2º edição, 2004

PETERS, James F.,PEDRYCZ, Witold, **Engenharia de Software – Teoria e Prética.** Rio de Janeiro: Campus, 2001.

PFLEEGER, Shari Lawrence. **Engenharia de Software –Teoria e Prática**. 2ª edição. São Paulo: Pearson Prentice Hall, 2003.

PRESSMAN, R. S. **Engenharia de software**, Trad. 5.ed. São Paulo: Mc Graw Hill, 2002.

SILVA J. Odair da, Carlos A. Borges, Clenio F. Salviano, A. L. Sampaio, A. N. Crespo, e Ana C. Roullier, **"An ISO/IEC 15504-Based Software Process Improvement Project in a Small Brazilian Software Organization"**, in Proceedings of SPICE2003: The Joint ESA – Third International SPICE Conference on Process Assessment and Improvement, Noordwijk, The Netherlands, p. 137-139, March 2003

SILVA J. Odair da; Carlos Alberto Borges; Clênio F. Salviano; Adalberto N. Crespo; Ana Cristina Roullier**, "Aplicação da ISO/IEC TR 15504 na Melhoria do Processo de Desenvolvimento de Software de uma Pequena Empresa"**, em Anais do Simpros 2003: Simpósio Internacional de Melhoria de Processo de Software, ISBN 85-7359-326-1, Recife, Brasil, p. 287- 288, Novembro 2003.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Addison Wesley, 2003.

SOUZA, Rogéria Cristiane Gratão, "Características de Testabilidade nos Diagramas UML (*Unified Modeling Language*): Apoio aos Testes de Sistemas de Software Orientados a Objetos", São Paulo, 2003.

THIOLLENT, Michel, **Metodologia da Pesquisa-ação**. Cortez Editora, 2004, 108p.

VILLAS BOAS, André Luiz de Castro, **Gestão de configuração para teste de software** / André Luiz de Castro Villas Boas.--Campinas, SP:[s.n.], 20

APÊNDICE A – Plano de Teste

Logotipo do
cliente

<Nome do Produto ou Serviço >

Cliente: <*Nome do Cliente*>

<Código do Projeto - Nome do Projeto>
PLANO DE TESTE

Versão <X.X>

Responsável pelo Documento: <*nome/função/e-mail* >

Logotipo da
Empresa

<nome, endereço, telefone da empresa>

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
dd/mm/aaaa	x.x		

1. INTRODUÇÃO

<Esta seção descreve de forma geral o plano de teste do produto ou serviço como um todo que está sendo construído.>

Este documento tem por finalidade descrever o plano geral das atividades de teste do projeto <nome do projeto>, fornecendo aos testadores e desenvolvedores as informações necessárias para a realização dos testes que compõe o sistema em desenvolvimento.

A atividade de teste constituirá uma forma de avaliar e agregar qualidade ao produto, reduzir custos e trabalho, dando maior confiabilidade ao software.

Serão definidas neste documento as estratégias de testes a serem adotadas a cada etapa do desenvolvimento do software, bem como a execução e armazenamento dos resultados dos testes.

As atividades realizadas neste projeto estão de acordo com a Política de Gerenciamento do Processo de desenvolvimento da desta Empresa.

2. ITENS QUE SERÃO TESTADOS

Os principais alvos a serem testados contemplados nesse plano serão:

Unidades de código – O código utilizado para implementar o software é o principal alvo do processo de teste. Os códigos desenvolvidos em uma linguagem de programação devem estar de acordo com requisitos e projeto do sistema. De maneira indireta também são testados os scripts unitários, de integração automáticos utilizados para validar o sistema de acordo com a arquitetura. De maneira geral, qualquer implementação que interaja com o sistema;

Sistema completo – Nos testes de integração e de aceitação o software permitirá o teste da interconexão entre os módulos e a funcionalidade final do software de acordo com os requisitos.

Requisitos e arquitetura – Os documentos utilizados como referência para os testes, comumente requisitos e arquitetura, também são indiretamente alvos dos testes, pois inconsistências relacionadas a esses documentos são naturalmente identificadas durante os testes.

3. ITENS QUE NÃO SERÃO TESTADOS

Esta seção indica as características que não serão considerados neste plano e as razões que justificam esta decisão.

4. O PROCESSO

Nesta seção será apresentado o fluxo de aplicação do processo de teste.

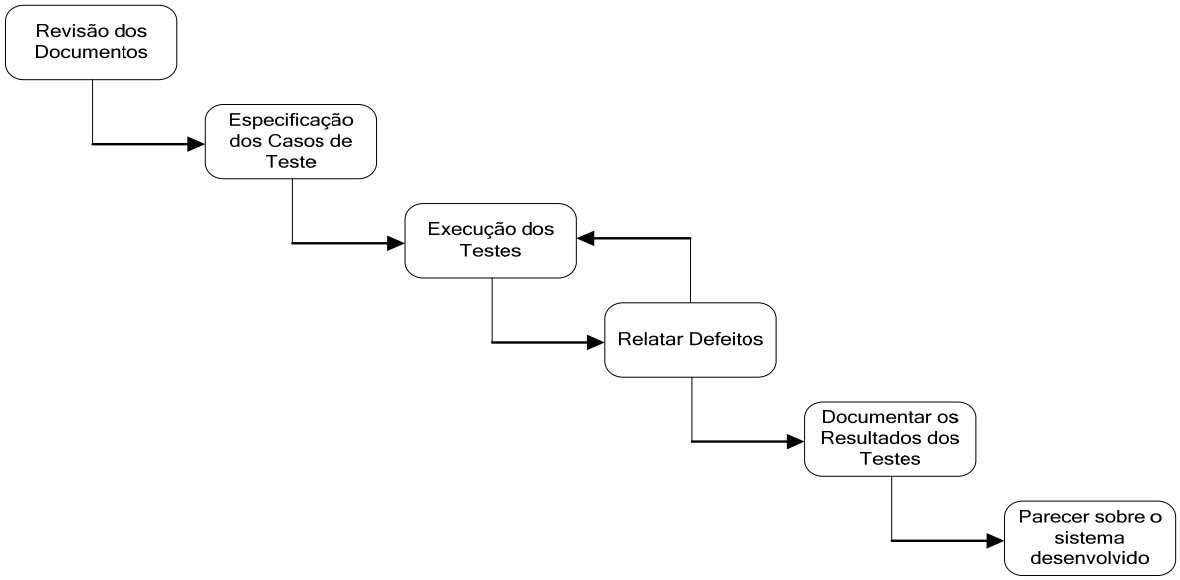


Figura 1 – Fluxo de execução dos testes

5. ESTRATÉGIA DE TESTE

Para o projeto <*Nome do Projeto*>, serão adotadas as seguintes estratégias:

- Teste de Unidade,
- Teste de Integração e
- Teste de Sistema.

Essas estratégias serão aplicadas conforme figura abaixo:

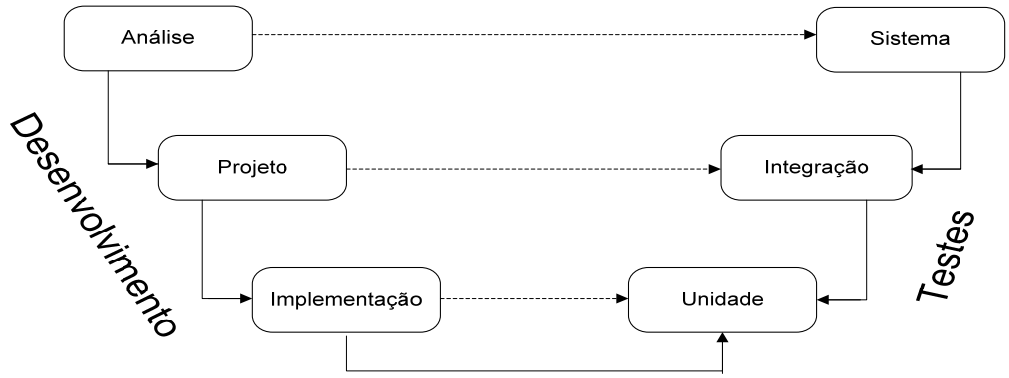


Figura 2 – Estratégia de Teste

Para a fase de implementação serão aplicados testes de unidade os quais contemplarão os testes de funcionalidades, testes de caixa preta, teste da base de dados, teste de aderência a padrões, teste de regressão (uma vez que esse é executado à medida que alterações, corretivas ou não, são aplicadas).

O teste de integração se aplica aos módulos e às interfaces existentes entre esses módulos do sistema. Deve-se realizar os testes de interface, funcionais (caixa-preta), da base de dados, de regressão nas interfaces entre os módulos.

O teste de sistema não está no escopo deste plano de teste, pois será aplicado numa etapa futura do processo, chamada Teste e Homologação.

5.1 Revisões

Objetivo – descobrir antecipadamente erros, de modo que eles não se propaguem para o passo seguinte do processo de desenvolvimento do software. A fim de garantir que erros ou inconsistência no planejamento do desenvolvido serão corrigidos ainda na fase de documentação.

Responsável – o Engenheiro de Teste deve executar as revisões em todos os documentos abaixo mencionados a fim de garantir que o que será desenvolvido obedece à solicitação do cliente.

Artefatos –

- **Entrada –** todos os documentos apresentados na tabela abaixo;
- **Saída –** relatórios gerados que informam os incidentes e inconsistências detectadas nos documentos.

Documento	Responsável
Documento de Requisito	<Nome do Responsável>
DCU XXX	
DCU XXY	

5.2 Teste de Unidade

Objetivo - identificar erros em procedimentos e funções. A partir do código-fonte são testadas as condições limites de cada unidade e os caminhos no fluxo de execução do software. O mecanismo interno do sistema será testado para garantir a qualidade de cada instrução do programa e sua função;

Responsável – o próprio Engenheiro de Software testará o requisito implementado antes que estes estejam integrados entre si para formar o sistema completo;

Execução – <a ser definido pelo responsável por este documento>;

Técnica – Automática e/ou Manual;

Artefatos –

- **Entrada –** Documentos de Casos de Uso;
- **Saída –** Documento de Caso de Teste.

Documento	Desenvolvedor	Testador
DCU XXX		
DCU XXY		

5.3 Teste de Integração

Objetivo – montar e integrar os módulos básicos a fim de fornecer um pacote de software. São testes de progressão onde os elementos de software ou de hadware, ou ambos são combinados e testados para avaliar suas interações, até que o sistema esteja integrado;

Responsável – Engenheiro de Teste analisando-se a conformidade de todos os casos de testes. Se houver alguma pendência a mesma deverá ser reportada à Ferramenta Mantis;

Execução – <a ser definido pelo responsável por este documento>;

Técnica – Manual e/ou Automática

Artefatos –

- **Entrada –** Documento de Caso de Teste, Matriz de Rastreabilidade contida no Documento de Requisitos e Plano de Teste de Integração.
- **Saída –** Planilha de Pendências.

Para uma melhor verificação da integração entre os módulos deverão ser incluídas no Plano de Integração as dependências entre os módulos a serem testados, bem como o resultado dos testes parciais e de integração. Deve se ainda elaborar um Plano de Integração para cada iteração do projeto.

5.4 Teste de Sistema

Objetivo – descobrir erros de funções e características de desempenho. Após a validação do software, um conjunto de especificações são analisados e combinados com elementos de sistemas como: hardware, pessoas, banco de dados, e outros. Todo o conjunto de elementos tem que combinar-se entre si adequadamente e verificar se suas funções e características de desempenho são atingidas;

Responsável – executados pelos engenheiros de teste, testes manuais, que tem como vantagem principal ser exploratório, o que antecipará problemas encontrados pelo cliente nos testes de aceitação;

Execução – os testes de sistema devem focar a iteração entre os componentes do sistema. Para isso serão realizados testes sistemáticos na interface e unidades para explorar a iteração dos componentes nos trechos de código dependentes do funcionamento de vários componentes e subsistemas integrados;

Artefatos –

- **Entrada –** são testes que usam os requisitos e especificações de requisitos para testar a interação dos vários módulos integrados do sistema. Simulando iterações reais dos usuários;
- **Saída –** relatórios de defeitos detectados.

5.4.1 Teste de Aceitação do Sistema

Os Testes de Aceitação devem ser realizados pelos clientes na liberação de cada release externo onde o cliente será convidado a interagir livremente pelo sistema a procura de erros ao mesmo tempo em que verifica os requisitos do sistema.

Determina se a implementação está em conformidade com o documento alvo (descrição de um sistema a ser implementado, que neste caso é construído na fase de análise de requisitos) ou não. Pequenas correções ainda podem ser feitas para ajustar características de sistema.

O Teste de aceitação será realizado pelo usuário final no ambiente de produção, sem a presença ou controle do desenvolvedor.

6. DOCUMENTANDO DEFEITOS

Quando forem identificados erros em quaisquer artefatos do software deve ser aberto um registro na ferramenta Mantis, como definido pelo processo, indicando o item de teste que precisa ser corrigido.

A identificação do defeito encontrado deve ser classificada de acordo com os procedimentos da Gerência da Configuração segundo sua gravidade e prioridade para que o desenvolvedor possa realizar uma análise sobre o defeito encontrado para corrigi-lo, quando necessário. A identificação do defeito deve ser documentada no relatório dos testes que falham. Para mais detalhes como documentar um defeito encontrado e trabalhar nesse defeito veja.

Para mais detalhes como documentar um defeito encontrado e trabalhar nesse defeito veja o manual de uso padrão da ferramenta Mantis.

6.1 Classificação e Prioridade dos Erros

Uma vez identificado algum erro sobre o requisito, deve-se proceder com a classificação do mesmo. Para tanto deve-se utilizar as informações da tabela a seguir.

Categoria	Descrição	Prioridade
1. Travamento	Os defeitos resultam em falhas do sistema ou de parte dele e não existe outra possibilidade, ou seja, o software pára completamente.	Resolver imediatamente
2. Obstáculo	Os defeitos resultam em falhas do sistema, entretanto existem alternativas de processo (manuais, por exemplo) que produzirão os resultados desejados.	Dar atenção alta
3. Pequeno	Os defeitos não resultam em falhas mas produzem resultados inconsistentes, incompletos ou incorretos ou prejudicam a usabilidade do software.	Fila normal
4. Mínimo	Os defeitos não causam uma falha, não prejudicam a usabilidade e os resultados do processamento desejado são obtidos contornando-se o problema.	Baixa Prioridade
5. Trivial	O defeito resulta de uma requisição de alteração ou melhoria, que pode ser indeferida.	Deferir ou não (longo prazo)

7. APROVAÇÃO

Para cada iteração do ciclo de vida do software devem ser definidos os critérios de aceitação para os testes que serão realizados de acordo com medidas de qualidade predefinidas.

<Abaixo segue um exemplo hipotético de planejamento e aceitação dos testes.>

Um teste passa quando todos os procedimentos são executados com sucesso. Falha quando ocorre uma divergência entre a saída produzida pelo sistema e a saída esperada descrita na verificação do caso de teste. Fica bloqueado quando as precondições não podem ser satisfeitas durante a execução.

7.1 Critérios de Sucesso

Item	Verificar	Aprovar
Funcionalidade	Deve ser verificada a presença de todas as funções mencionadas; a execução correta destas funções; a ausência de contradições entre a descrição do produto e a documentação do usuário.	Só será aprovado requisitos 100% implementados e sem erros graves.
Confiabilidade	O usuário deve manter o controle do produto, sem corromper ou perder dados, mesmo que a capacidade declara seja explorada até os limites ou fora deles, se uma entrada incorreta é efetuada, ou ainda se instruções explícitas na documentação são	Só serão aprovados requisitos que respeitem 100% do descrito ao lado.

	violadas.	
Usabilidade	A comunicação entre o programa e o usuário deve ser de fácil entendimento, através das entradas de dados, mensagens e apresentação dos resultados, utilizando um vocabulário apropriado, representações gráficas e funções de auxílio (help), entre outras; o programa também deve proporcionar uma apresentação e organização que facilite uma visão geral das informações, além de procedimentos operacionais que auxiliem, por exemplo, a reversão de uma função executada e o uso de recursos de hipertexto em funções de auxílio, entre outras.	Só serão aprovados requisitos que respeitem 100% do especificado ao lado.

8. CRONOGRAMA

<Esta seção apresenta o cronograma para as atividades de teste do projeto. No cronograma devem constar as atividades, marcos, dependências e recursos humanos alocados. Caso a fase de planejamento do desenvolvimento do projeto já contenha um cronograma relativo às atividades de teste, basta informar qual o documento ou ferramenta que o contêm. >

9. REFERÊNCIAS

<Esta seção deve prover uma lista de todos os documentos relacionados a este documento. >

Representante do contratando	**Representante da contratante**
Testemunha 1	**Testemunha 2**

APÊNDICE B – Documento de Caso de Teste

Logotipo do cliente

<Cód. Projeto - Nome do Projeto >
Cliente: *<Nome do Cliente>*

CASO DE TESTE GERAL
Versão 0.1

Logotipo da Empresa

Responsável pela Elaboração do Documento: *<Nome do Responsável>*
Data da Elaboração: *<dd/mm/aaaa>*

RF00 – <Nome do Requisito>

---

1. HISTÓRICO DA EXECUÇÃO DOS TESTES

Deve ser registrado: a data o responsável pela execução dos testes, indicando seu cargo na equipe do projeto, a breve descrição da situação e o tempo de execução dos testes.

Data	Executor/Cargo	Descrição	Duração (horas)
dd/mm/aaaa	*<informar o nome e o cargo do executor dos testes>*	*<descrição do status do requisito, ou seja, se foi encontrado erros, se espera reteste ou se foi aprovado. >*	*HH:MM:SS*

2. VISÃO GERAL DO DOCUMENTO

Este documento tem como objetivo ser um guia para a realização de testes de unidade para o Engenheiro de Software e para verificação do Engenheiro de Teste, além de ser verificado durante a fase de teste de integração do projeto.
Os testes a serem realizados deverão atender a quatro aspectos diferentes, a saber:

- Teste Funcional;
- Teste de Validação;
- Teste de Base de Dados;
- Teste de Integração.

3. ITENS A TESTAR

Nesta seção são identificados e descritos os itens de software e as características que serão exercitadas, deve-se ainda tratar testes relacionados a itens que se enquadrem as regras do negócio do requisito.

3.1 Interface

Nº	Descrição	Status
1.1	Os campos monetários, percentagem, peso etc, está dentro do padrão: <informar o padrão para os campos e se for necessário informar quais campos são>	☐ *OK*
1.2	As máscaras definidas foram implementadas para os campos específicos: <informar os campos e as máscaras especifica para cada qual>	☐ *OK*
1.3	As mensagens informativas para o usuário estão de acordo com o padrão estabelecido especificado a seguir: - validação: <informar a mensagem padrão e quais os campos a enviam ao usuário> - de erro: <informar a mensagem padrão e quais os campos a enviam ao usuário> - de preenchimento obrigatório: <informar a mensagem padrão e quais os campos a enviam ao usuário> - etc.	☐ *OK*
1.4	Os campos obrigatórios estão sendo indicados por <informar se o campo é indicado por imagem, frase etc de preenchimento obrigatório>.	☐ *OK*
1.5	Os seguintes botões foram implementados de acordo com o nome e padrão preestabelecido <se necessário indicar o nome e padrão >.	☐ *OK*

3.2 Implementação

Nº	Descrição	Status
2.1	Todos os campos descritos no DCU foram implementados corretamente, de acordo com o descrito a seguir: <informar campos tipo e tamanho especifico>	☐ *OK*
2.2	Os campos aceitam apenas as entradas de dados definidas, ou seja: <informar os campos e o tipo de entrada aceita (letras/números/ambos)>	☐ *OK*
2.3	As mensagens informativas estão se referindo corretamente ao atributo e/ ou item cadastrado.	☐ *OK*
2.4	Os campos não obrigatórios, ou seja, que podem ser nulos, estão sendo tratados corretamente. <se necessário informar qual deve ser o tratamento padrão>	☐ *OK*
2.5	Os campos com combobox possuem default, ou possuem mensagens informando ao usuário o seu preenchimento. <Se existir mensagem informe os campos e as mensagens estabelecidas junto com o padrão estabelecido (se é ordenado em ordem alfabética, ou em ordem crescente ou decrescente etc.)>	☐ *OK*
2.6	As opções radiobox devem vir marcadas/desmarcadas. <Em caso de vir com opção marcada especificar qual deve vir marcada>	☐ *OK*
2.7	Os campos <se necessário informar quais são os campos com esse tipo de estrutura> com opções radiobox funcionam corretamente, ou seja, quando uma opção é selecionada, automaticamente é desmarcada a opção que estava marcada.	☐ *OK*
2.8	Os campos com opções checkbox devem vir marcadas/desmarcadas/com apenas uma opção marcada. Em caso de uma opção marcada, especificar qual deve vir marcada.	☐ *OK*
2.9	As mensagens de preenchimento obrigatório dos campos do requisito aparecem após confirmação do cadastro/automaticamente. E aparecem em caixa de diálogo ou em frases em frente ao campo.	☐ *OK*
2.10	Os seguintes link's <informar quais os link's> foram implementados, e carregam as seguintes páginas <informar as páginas>	☐ *OK*
2.11	Nos seguintes campos <informar os campos> deve ser executado a validação.	☐ *OK*
2.12	O fluxo da validação é <se ela deve ser feita automaticamente, ou após conclusão do cadastro>.	☐ *OK*
2.13	O fluxo da mensagem é <se ela deve ser feita automaticamente, ou após conclusão do cadastro>.	☐ *OK*
2.14	O fluxo do sistema após a seleção dos botões é <informar qual o botão e o destino após sua seleção>.	☐ *OK*
2.15	A função do botão é <informar o botão e sua função>.	☐ *OK*
2.16	Os campos habilitados apenas para visualização, ou seja, desabilitados para entrada de dados foram implementados corretamente.	☐ *OK*

3.3 Banco de Dados

Nº	Descrição	Status

3.1	Os valores limites para os campos (variáveis com intervalo máximo e mínimo) estão de acordo com o estabelecido.	☐ *OK*
3.2	Os valores cadastrados no BD para os campos com opção radiobox são <informar qual o valor será cadastrado no BD caso exista apenas um radiobox, caso existam vários radiobox para o campo informa quais os valores de cada um caso seja cadastrado no BD>.	☐ *OK*
3.3	Os valores cadastrados no BD para os campos com opção checkbox são: <informar qual o valor será cadastrado no BD caso exista apenas um checxbox, caso existam vários checkbox para o campo informa quais os valores de cada um caso seja cadastrado no BD>	☐ *OK*
3.4	Os dados cadastrados estão sendo inseridos corretamente na tabela <informar qual a tabela especifica> do BD.	☐ *OK*
3.5	O sistema verifica corretamente os dados já cadastrados no BD, afim de não permitir a inserção de dados duplicados <informar quais campos não devem permitir o cadastro de dados duplicados>.	☐ *OK*

4. RELATÓRIO-RESUMO DE ERROS

Nesta seção deve ser apresentado de forma resumida os resultados das atividades para posterior avaliação no fim de cada iteração e do projeto como um todo.

Detecção do Erro	Item e Registro	Classificação	Data de Aprovação
dd/mm/aaaa	<Nº e registro no mantis>	*<verifique no DRE o critério de classificação do erro>*	*<informe a data em que foi verificado a correção do erro> dd/mm/aaaa*