RAFAEL SILVA BARRETO

ESTUDO E PROPOSTA DE UM PROCESSO DE TESTE PARA UMA COOPERATIVA DE SOFTWARE LIVRE

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
2008

RAFAEL SILVA BARRETO

ESTUDO E PROPOSTA DE UM PROCESSO DE TESTE PARA UMA COOPERATIVA DE SOFTWARE LIVRE

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. D.Sc. José Monserrat Neto

LAVRAS
MINAS GERAIS – BRASIL
2008

Ficha Catalográfica

Barreto, Rafael Silva

Estudo e Proposta de um Processo de Teste para uma Cooperativa de Software Livre / Rafael Silva Barreto. Lavras – Minas Gerais, 2008. 68p : il.

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

1. Informática, 2. Engenharia de Software, 3. Teste de Software. I. BARRETO, R. S.. II. Universidade Federal de Lavras. III. Título.

RAFAEL SILVA BARRETO

ESTUDO E PROPOSTA DE UM PROCESSO DE TESTE PARA UMA COOPERATIVA DE SOFTWARE LIVRE

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

_____
Prof. D.Sc. Heitor Augustus Xavier Costa

_____
Profª. M.Sc. Juliana Galvani Greghi

_____
Prof. D.Sc. José Monserrat Neto
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

Dedico este trabalho aos meus pais, Mario e Maria Aparecida, aos meus irmãos Rodrigo e Ruan, por fazerem parte dessa família que eu amo.

AGRADECIMENTOS

Agradeço,

A Deus, primeiramente, por ter iluminado o caminho que percorri para chegar até aqui.

A toda minha família, por ter acreditado em mim e sempre me incentivado para nunca desistir. Podem contar comigo para o que for.

Aos meus pais, pelo amor, carinho e apoio em todos os momentos da minha vida. Obrigado por todos os ensinamentos passados, lições e pela confiança depositada em mim.

Aos meus irmãos, pelo companheirismo, nos momentos de alegria e tristeza estiveram sempre do meu lado.

Aos meus tios, avós, padrastos, primos, por sempre me confortar com palavras positivas.

Aos meus amigos, de todas as etapas da minha vida, todos contribuíram de alguma forma para me ajudar a crescer como pessoa. Amizades verdadeiras são pra sempre.

Aos companheiros das repúblicas onde morei, em Viçosa e Lavras, nos momentos mais difíceis dessa etapa, estiveram sempre do meu lado.

Aos professores, do primário à universidade, que formaram a base do meu conhecimento para o lado acadêmico e para a outras etapas da vida. Obrigado, aos professores do Departamento de Ciência da Computação da UFLA e em especial, ao Professor Monserrat, que confiou na minha capacidade e ajudou na concretização desse trabalho.

Por último, não menos importante, aos companheiros da TecnoLivre, que me ajudaram no aprendizado, me deram oportunidades e, principalmente, pelo companheirismo dentro e fora do ambiente de trabalho.

ESTUDO E PROPOSTA DE UM PROCESSO DE TESTE PARA UMA COOPERATIVA DE SOFTWARE LIVRE

RESUMO

Teste vem sendo reconhecido cada vez mais como atividade chave para o desenvolvimento de software de qualidade. A definição de um processo de teste contribui para que este desenvolvimento seja realizado de forma organizada e eficiente. Neste trabalho, apresenta-se um estudo sobre teste, processo de teste e qualidade de software, e propõe-se um processo de teste para uma cooperativa que desenvolve software - TecnoLivre. A proposta foi elaborada em consonância com o processo de desenvolvimento de software implantado no empreendimento.

Palavras-chave: Processo de Teste, Estratégias de Teste, Técnicas de Teste, Qualidade de Software.

STUDIES AND PROPOSE OF SOFTWARE TESTING PROCESS IN A COOPERATIVE OF FREE SOFTWARE

ABSTRACT

Testing has been increasingly acknowleded as a key activity for quality software development. The definition of a test process contributes to making this development more organized and efficient. This work examines studies about test, test process and software quality, and proposes a test process for a cooperative which develops software - TecnoLivre. This proposal was made in line with the software development process used by the cooperative.

Keywords: *Testing Process, Testing Strategies, Testing methods, Software Quality.*

SUMÁRIO

LISTA DE FIGURAS....ix

LISTA DE TABELAS....x

1 INTRODUÇÃO....1
1.1 Metodologia....2
1.2 Estrutura do Trabalho....3

2 TESTE DE SOFTWARE....4
2.1 Definição de Teste....4
2.2 Estratégias de Teste de Software....5
2.2.1 Teste de Unidade....8
2.2.2 Teste de Integração....8
2.2.3 Teste de Validação....10
2.2.4 Teste de Sistema....11
2.3 Técnicas de Teste de Software....12
2.3.1 Teste Estrutural....13
2.3.2 Teste Funcional....14
2.3.3 Teste Orientado a Objetos....15
2.4 Considerações Finais....17

3 PROCESSO DE TESTE....18
3.1 Qualidade de Software....19
3.1.1 Revisões....20
3.1.2 Garantia da Qualidade de Software....21
3.2 Qualidade no Processo de Teste....23
3.3 NBR ISO/IEC 12119....23
3.3.1 Requisitos de Qualidade....24
3.3.2 Instruções para Teste....26
3.4 Norma IEEE 829....28
3.5 Etapas do Processo de Teste....30
3.6 Implantação de um Processo de Teste....32
3.7 Considerações Finais....34

4 PROCESSO DE TESTE DA TECNOLIVRE....35
4.1 Concepção do Processo....35
4.2 Definição do Projeto....40
4.2.1 Os Papéis....40
4.2.2 Atividades do Processo....41
4.3 Fluxograma do Processo de Desenvolvimento....51
4.4 Considerações Finais....52

5 CONCLUSÃO.....53
5.1 Trabalhos Futuros.....53
6 REFERÊNCIAS BIBLIOGRÁFICAS.....54

LISTA DE FIGURAS

Figura 2.1: Estratégia de Teste (Fonte: Pressman, 2006).....7
Figura 3.1: Qualidade Baseada em Processos (Fonte: Sommerville, 2003).....20
Figura 3.2: Relacionamento entre os Documentos de Teste (Fonte: Crespo 2004).....30
Figura 3.3: Ambientes de Teste (Fonte: Rios & Moreira, 2006).....33
Figura 4.1: Processo de Desenvolvimento de Software (Fonte: Coelho, 2007).....36
Figura 4.2: Ciclo de Vida do Processo de Teste (FONTE: Rios & Moreira, 2006).....38
Figura 4.3: Nova Estrutura Organizacional Para os Projetos.....41
Figura 4.4: Atividades do Processo de Teste.....42
Figura 4.5: Processo de Desenvolvimento de Software (Fonte: Coelho 2007, adaptado).....51

LISTA DE TABELAS

Tabela 3.1: Tipos de Revisão (Fonte: Sommerville, 2003).....21
Tabela 3.2: Atividade de Garantia de Qualidade (Fonte: Pádua, 2003).....22
Tabela 3.3: Perfil de um Profissional de Teste (Fonte: Rios & Moreira, 2006).....32
Tabela 4.1: Fase do Processo de Desenvolvimento e Atividade de Teste Correspondente.....42

1 INTRODUÇÃO

No mundo moderno, o sistema de informação passou a ser um grande aliado das empresas no apoio aos negócios e o software é uma importante ferramenta de tecnologia utilizada em sistemas nas mais diversas áreas da sociedade. Isso gera grande demanda para as empresas desenvolvedoras de software, ao mesmo tempo, gera forte pressão por parte dos clientes para produzir software no menor tempo possível.

Além da demanda por software construídos rapidamente, há uma exigência cada vez maior por software de qualidade. Não podem haver falhas em sistemas como controladores de tráfego aéreo, gerenciadores hospitalares, além de outros tipos que envolvem vidas humanas, negócios milionários de empresas, ou pesquisas científicas. Por isso, é necessário que o software utilizado em tais sistemas tenha cada vez mais qualidade.

Organizações desenvolvedoras de software utilizam alguns artifícios para conseguir qualidade na sua produção. Um deles é utilizar um processo de desenvolvimento de software. Um processo bem definido e documentado contribui para que o software seja produzido com qualidade.

Uma das atividades do processo de grande importância na garantia de qualidade é o teste de software. Muito se discute sobre a prática de testar o software apenas no fim do processo de desenvolvimento. Erros[1] durante a produção podem ocorrer em qualquer etapa do processo, sendo que quanto antes detectá-los, menor será o gasto final e o tempo utilizado para a correção. Por conta disso, estudos apontam que realizar atividades de teste ao longo do o processo de desenvolvimento de software pode ser eficaz, reduzindo custos na produção (Myers, 2004; Pressman, 2006; Delamaro 2007).

Para conseguir bons resultados com teste, as organizações desenvolvedoras de software devem utilizar algum processo de teste. Assim como o processo de desenvolvimento é um auxílio para a produção, o processo de teste é uma forma de extrema valia para organizar as atividades de teste.

Esse trabalho tem como objetivo realizar um estudo e propor um processo de teste para uma cooperativa que desenvolve software livre, tendo como base os conceitos gerais de engenharia de software, a definição de processo de teste e o seu propósito especificado.

---

1 Erros, falhas e defeitos serão tratados como sinônimos no trabalho, para facilitar o entendimento. Na área de Qualidade de Software, erro é um engano cometido pelo ser humano, defeito é o resultado de um erro encontrado em um código ou em um documento e falha é o resultado ou manifestação de um ou mais defeitos.

A cooperativa em questão é a TecnoLivre - Cooperativa de Tecnologia e Soluções Livres, da qual faço parte como membro fundador. Esta organização está sediada na cidade de Lavras – MG. A organização atua em diversas áreas de TI, composta por profissionais com bacharelado em Ciência da Computação e com alunos deste curso. Uma de suas principais atividades é o desenvolvimento de sistemas, seja *desktop* ou *web*.

A TecnoLivre utiliza processo de desenvolvimento de software proposto por Coelho (2007). O processo é simples e objetivo, focado na documentação, na definição dos papéis envolvidos e com boa interação com os clientes. Porém, a fase de teste não vem sendo realizada como foi definida no processo.

Tem-se como principal motivação para este trabalho a necessidade de melhorar a atividade de teste durante os projetos da cooperativa. Definindo um processo de teste, pode ser uma forma de conseguir tal melhoria.

A proposta de processo de teste foi baseada nas características do processo de desenvolvimento de software implantado na cooperativa, citadas anteriormente, e em metodologias de processo de teste encontrados na literatura. O processo envolve mudança na estrutura organizacional da cooperativa e nas atividades de teste, adaptadas no ciclo do processo de desenvolvimento, de forma que ele traga o menor impacto possível para a organização.

1.1 Metodologia

O trabalho consistiu no aprofundamento sobre o assunto do tema proposto, sendo feito um estudo e uma análise sobre processos, estratégias e técnicas de teste. De acordo com Jung (2004), o tipo de pesquisa utilizada se classifica como bibliográfica, uma vez que tem como finalidade o maior entendimento sobre o assunto e aprimoramento de idéias. A pesquisa bibliográfica tem por finalidade conhecer as diferentes formas de contribuição científica que se realizaram sobre determinado assunto ou fenômeno.

Quanto aos seus objetivos, trata-se de uma pesquisa de caráter exploratório, pois visa à proposição de um novo processo de teste de software a partir de modelos de processos existentes. Segundo Jung (2004), uma pesquisa de caráter exploratório busca a descoberta de novas teorias e práticas que modifiquem as existentes.

A pesquisa baseou-se em referências bibliográficas, tais como livros e artigos científicos e pela Internet, usando definições sobre processos, estratégias e técnicas de teste e modelos de processo de teste de software existentes.

1.2 Estrutura do Trabalho

O presente trabalho foi dividido da seguinte forma: no Capítulo 2, são apresentados o conceito da atividade de teste e as estratégias e as técnicas utilizadas para essa atividade. No Capítulo 3, são apresentados a importância da qualidade de software para uma empresa desenvolvedora de software e o papel do processo de teste para garantia da qualidade. Além disso, são apresentadas as etapas do processo de teste e as normas nas quais o processo resultante deste trabalho se baseou. No Capítulo 4, é apresentada a proposta de processo de teste, com o embasamento para a sua elaboração, a estrutura da cooperativa e a definição das atividades de teste. No Capítulo 5, são apresentadas a conclusão do trabalho e algumas propostas de trabalho futuro.

2 TESTE DE SOFTWARE

A construção de um software não é uma tarefa simples, pelo contrário, pode se tornar bastante complexa, dependendo das características e das dimensões do sistema a ser criado. Por isso, problemas podem ocorrer durante o seu desenvolvimento, obtendo um produto diferente do que era esperado.

Segundo Delamaro *et al.* (2007), muitos fatores podem ser identificados como causas de tais problemas, mas a maioria deles tem uma única origem: erro humano. Como a maioria das atividades de engenharia, a construção de software depende principalmente da habilidade, da interpretação e da execução das pessoas que o desenvolve. Por isso, erros acabam surgindo, mesmo com a utilização de métodos e de ferramentas de engenharia de software.

A fim de se evitar que tais problemas impactem de forma significativa e negativa o desenvolvimento de um software, surgiram algumas atividades com a finalidade de garantir que o produto final seja aquele especificado. Uma delas é a atividade de teste.

2.1 Definição de Teste

A atividade teste de software tornou-se uma das mais importantes durante o desenvolvimento de um software. Por Myers (2004), teste de software é um processo ou uma série de processos concebidos para certificar que um programa faça exatamente aquilo para o qual foi projetado para realizar e que não faça nada não intencional. Um software deve ser previsível e consistente, e não deve oferecer quaisquer surpresas para os usuários.

Myers (2004) enumerou algumas regras que podem servir como objetivos de teste:

- Teste é um processo de execução de um programa com a finalidade de encontrar erro;
- Um bom caso de teste é aquele que tem alta probabilidade de encontrar um erro, ainda não descoberto;
- Um teste bem-sucedido é aquele que descobre um erro, ainda não descoberto.

Tais regras vão de encontro ao pensamento que se tinha quando surgiram as primeiras atividades de teste, na década de 70, pois se dizia que o teste servia apenas para demonstrar que o programa não continha erros.

A atividade de teste de um software se inicia no momento em que se inicia o seu desenvolvimento e perdura até o fim do ciclo de desenvolvimento. Atualmente, com a importância dada a esta àrea da engenharia de software, faz-se necessário ter um processo de teste bem definido e organizado para garantir que o produto final seja de qualidade.

Segundo Pressman (2006), se o teste é conduzido ao acaso, tempo é desperdiçado, esforço desnecessário é despendido e, ainda pior, erros se infiltram sem serem descobertos. Assim, é aconselhável estabelecer uma estratégia sistemática para o teste de software.

2.2 Estratégias de Teste de Software

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 bem-sucedida de software. A estratégia fornece um roteiro que descreve os passos a serem conduzidos como parte do teste, quando esses passos são planejados e depois executados, e quanto de esforço, tempo e recursos serão necessários. Assim, qualquer estratégia de teste deve incorporar planejamento de teste, projeto de casos de teste, execução de teste e o resultado da coleta e da avaliação de dados (PRESSMAN, 2006).

Um ponto crucial na atividade de teste, independentemente da fase, é o projeto e/ou a avaliação da qualidade de um determinado conjunto de casos de teste, utilizado para o teste de um produto, pois, em geral, é impraticável utilizar todo o domínio de dados de entrada para avaliar os aspectos funcionais e operacionais de um produto em teste. O objetivo é utilizar um conjunto de casos de teste que tenham alta probabilidade de encontrar a maioria dos defeitos com um mínimo de tempo e esforço, por questões de produtividade.

Algumas estratégias de teste de software propostas na literatura fornecem ao desenvolvedor de software um gabarito de testes e as seguintes características genéricas (Pressman, 2006; Sommerville, 2003; Myers, 2004) :

- Para realizar teste efetivo, uma equipe de teste de software deve conduzir revisões formais, e, ao fazer isso, muitos erros serão eliminados antes do início do teste;
- O teste começa no nível de componente e prossegue em direção à integração do sistema baseado em computador;
- Diferentes técnicas de teste são adequadas em diferentes momentos, cada uma tendo sua própria característica e finalidade de encontrar determinado tipo de erro;

- O teste é conduzido pelo desenvolvedor do software e/ou por um grupo de teste independente, em caso de projetos maiores.

A atividade de teste fez surgir, por necessidade, um novo profissional na área de Engenharia de Software, o Engenheiro de Teste. Cabe a ele a função de criar desde as estratégias de teste da empresa até os planos de testes, em que são definidos o que cada participante de cada projeto fará para a realização do teste.

O desenvolvedor também deverá participar da atividade, fazendo testes de mais baixo nível, como verificar pedaços de códigos ou testar a integração dos módulos do sistema. No entanto, como afirma Pressman (2006), do ponto de vista psicológico, a análise, o projeto e a codificação são tarefas construtivas. Assim, no pensamento de um construtor, como um Engenheiro de Software ou um Programador, o teste pode ser visto como destrutivo. Desta forma, o desenvolvedor do sistema tende a seguir suavemente, projetando e executando testes que vão demonstrar que o programa funciona, ao invés de encontrar erros.

Por isso, em uma organização desenvolvedora de software, deve existir a figura do Engenheiro de Teste ou, em alguns casos, deve existir um grupo independente de teste. O teste realizado por pessoas contratadas para encontrar erros remove o conflito de interesses que poderia estar presente. De qualquer modo, o engenheiro de software e os desenvolvedores continuarão a acompanhar os testes em conjunto com o engenheiro de teste ou o grupo encarregado, para garantir que os testes rigorosos sejam conduzidos, além da necessidade da correção imediata do erro, caso seja descoberto algum.

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 a serem conduzidos (SOMMERVILLE, 2003).

Pressman (2006) afirma que uma estratégia de teste de software deve acomodar testes de baixo nível, necessários para verificar se um pequeno segmento de código-fonte foi corretamente implementado, bem como testes de alto nível, que validam as principais funções do sistema com base nos requisitos do cliente. Uma estratégia deve fornecer diretrizes para o profissional encarregado do teste e um conjunto de referências para o gerente.

Com a evolução da atividade de teste de software, diferentes estratégias podem ser adotadas para diferentes tipos de software. Métodos e técnicas de teste mais específicos são usados para testar software de acordo com suas características, isto é, existem, por exemplo,

técnicas de teste apropriadas para arquiteturas convencionais, outras para arquiteturas orientadas a objetos, e outras para arquiteturas orientadas a aspectos.

De forma generalizada, pode-se montar uma estratégia de teste em etapas. Pressman (2006) afirma que, assim como o processo de engenharia de software é visto como uma espiral, uma estratégia de teste pode ser colocada no contexto da espiral, tal como mostrado na Figura 2.1.

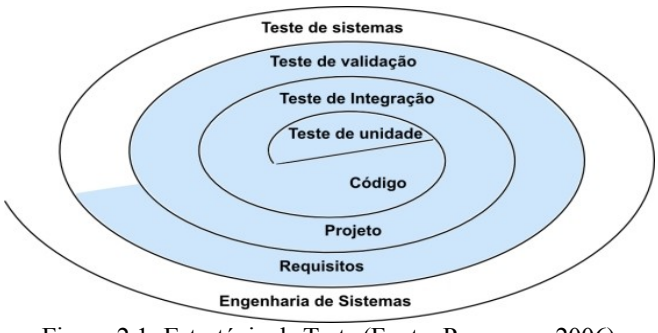

Figura 2.1: Estratégia de Teste (Fonte: Pressman, 2006)

Na engenharia de software, move-se para dentro na espiral, ao longo de voltas que diminuem o nível de abstração. Inicialmente, a engenharia de sistemas define o papel do software e realiza a análise de requisitos do software, em que são estabelecidos o domínio da informação, a função, o comportamento, o desempenho, as restrições e os critérios de validação para o software. Os passos seguintes são os de projeto e codificação.

Em uma estratégia de teste do software, o começo é no centro da espiral, com o teste de unidade, o qual se concentra em cada unidade do software. Movendo-se para fora, chega-se ao teste de integração, cujo foco está no projeto e na arquitetura do sistema. Progredindo na espiral, chega-se ao teste de validação, em que os requisitos estabelecidos são validados em confronto com o software recém construído. Por fim, tem-se o teste de sistema, em que o software e todo o sistema são verificados como um todo.

Algumas literaturas apresentam o teste de validação com o nome de teste de aceitação, colocando-o antes do teste do sistema (Rios & Moreira, 2006). Outros consideram os dois testes, sistema e validação, em uma mesma etapa (Delamaro *et al.*, 2007). Realizar primeiro o teste de validação ou o teste de sistema, parece ser uma questão ainda não definida, cabendo ao Engenheiro de Teste definir a seqüência que melhor atende a cada projeto.

A seguir serão explicados com detalhes cada uma das etapas de uma estratégia de teste.

2.2.1 Teste de Unidade

Teste de unidade é o estágio mais baixo na escala de testes e são aplicados nos menores componentes de código criados, visando garantir que estes atendem as especificações, em termos de características e funcionalidade. Os testes unitários verificam o funcionamento de um pedaço do sistema ou software isoladamente, ou que possam ser testados separadamente, podendo, inclusive, ser um programa ou um componente. Na grande maioria dos casos, estes testes são realizados pelos desenvolvedores (RIOS & MOREIRA, 2006).

Nessa etapa da estratégia de teste, espera-se encontrar erros relacionados a algoritmos incorretos ou mal implementados, estrutura de dados incorretas ou simples erros de programação, como erro de sintaxe. Nesse teste, não há necessidade de dispor do sistema totalmente finalizado, uma vez que as unidades do software são testadas separadamente, podendo o teste ser aplicado à medida que ocorre a implementação das unidades.

No contexto de orientação a objeto, o conceito do teste de unidade é alterado. Na visão de Pressman (2006), uma classe encapsulada é o foco do teste de unidade. Encapsulamento quer dizer que a classe e cada instância dela (objeto) empacotam os atributos (dados) e as operações (funções) que manipulam esses dados. No entanto, operação dentro de classes é a menor unidade estável. Como uma classe pode conter um certo número de operações diferentes e uma operação particular pode existir como parte de um certo número de classes diferentes, deve-se testar uma operação como parte de uma classe, e não mais isoladamente.

O teste de unidade se torna importante em uma estratégia, pois assim pode-se evitar que os possíveis erros, citados acima, apareçam em um estágio mais avançado do desenvolvimento do software, evitando gastos maiores de tempo e recursos.

2.2.2 Teste de Integração

Segundo Rios & Moreira (2006), o teste de integração é executado em uma combinação de componentes para verificar se eles funcionam corretamente juntos, ou seja, assegurar que as interfaces funcionem corretamente e que os dados são processados de forma correta, conforme as especificações. Componentes podem ser pedaços de códigos, módulos, aplicações distintas, clientes e servidores.

Estes testes podem ser feitos de forma incremental, ou seja, cada módulo ou componente é incluído seqüencialmente até que todos os casos de testes possam ser executados. Isso evita uma estratégia de teste chamada *big-bang*, tendencialmente utilizada,

cuja idéia é testar o software depois que todos os módulos ou componentes sejam integrados. Tal abordagem dificulta ainda mais a correção de um possível erro, pois o custo é menor ao corrigir um possível erro testando isoladamente um módulo ou componente.

Nesse tipo de teste é necessário um grande conhecimento das estruturas internas e das interações existentes entre as partes do sistema, por isso, o teste tende a ser executado pela própria equipe de desenvolvimento.

Duas estratégias de testes importantes podem ser utilizadas para a realização do teste de integração (Rios & Moreira, 2006; Pressman, 2006):

- **Integração ascendente (*bottom-up*):** os testes são realizados agrupando os componentes, programas, módulos ou subsistemas de mais baixo nível para formar novos módulos ou subsistemas de níveis superiores. A integração é feita ascendentemente pela hierarquia de controle até chegar ao módulo de controle principal. É bem utilizado para construção de sistemas de grande porte, pois, cada equipe de desenvolvimento é responsável pela criação de um subsistema;

- **Integração descendente (*top-down*):** é o inverso da estratégia anterior. Uma vantagem desta estratégia é alguns resultados poderem ser apresentados para os usuários antes de completada a construção de componentes, programas, módulos ou subsistemas de níveis mais baixos, desde que artifícios sejam criados para simular suas ausências. Um problema que pode surgir nesse tipo de integração é quando há necessidade de processamento de níveis mais baixos da hierarquia para testar adequadamente níveis superiores.

Para teste de integração em sistemas orientados a objetos, Pressman (2006) cita dois tipos de estratégias diferentes:

- **Teste baseado no caminho de execução (*thread-based testing*):** integra-se o conjunto de classes necessárias para responder uma entrada ou um evento do sistema. Cada caminho de execução é integrado e testado individualmente. O teste de regressão[2] é aplicado para garantir que efeito colateral não ocorra;

- **Teste baseado no uso (*use-based testing*):** o sistema é construído testando primeiro as classes independentes, que utilizam poucas classes servidoras. O passo seguinte é testar a camada de classes dependentes, que usam as classes independentes. A

2 O teste de regressão é uma técnica de teste aplicada em partes modificadas do software que haviam sido testadas anteriormente. Alterações realizadas, especialmente durante a manutenção, podem introduzir erros nas partes testadas.

seqüência de teste de camadas de classes dependentes continua até que todo o sistema esteja concluído.

2.2.3 Teste de Validação

O teste de validação começa no fim do teste de integração, quando componentes individuais foram testados, o software está completamente montado como um pacote e os erros de interface foram descobertos e corrigidos. Nesta etapa da estratégia de teste, não há mais distinção entre o software convencional e orientado a objetos. O teste focaliza ações visíveis ao usuário e saídas do sistema reconhecidas pelo usuário (PRESSMAN, 2006).

São validados os requisitos levantados na análise de requisitos do software, isto é, o teste é realizado para verificar se o sistema faz aquilo que foi solicitado. Esta pessoa pode ser um profissional da equipe de teste, que faz o papel de cliente, ou pode ser o próprio cliente instado a verificar a funcionalidade do sistema. De acordo com Pressman (2006), o documento de Especificação dos Requisitos do Software contém uma seção chamada Critérios de Validação. A informação contida nessa seção forma a base para a abordagem do teste de validação.

Quando o software é desenvolvido para atender um cliente, é comum serem feitos testes de aceitação pelo próprio cliente, podendo ser efetuado de uma forma mais informal ou planejada. Caso o software desenvolvido seja usado por vários clientes, torna-se impraticável a realização do teste de aceitação por todos. Os desenvolvedores utilizam dois processos:

- **Teste alfa:** é realizado no ambiente do desenvolvedor e praticado pelo usuário final. Este testa o software e o desenvolvedor observa as reações do usuário e registra possíveis erros e problemas de uso. Este tipo de teste é conduzido num ambiente controlado;

- **Teste beta:** esse tipo de teste é conduzido no ambiente do usuário, sem a interferência do desenvolvedor. O cliente registra os erros encontrados pelo usuário e os envia para o desenvolvedor em intervalos de tempo, para que possam ser solucionados.

Segundo Pressman (2006), após serem conduzidos cada caso de teste de validação, pode-se chegar a duas condições: i) a funcionalidade do sistema satisfaz as especificações e são aceitas; ou ii) um desvio das especificações é descoberto e uma lista de deficiências é criada. Desvios ou erros descobertos neste estágio do desenvolvimento do software, em um projeto, raramente podem ser corrigidos antes da entrega programada, pois demandará um custo maior

maior para a correção. Cabe então ao responsável negociar com o cliente para estabelecer uma solução para as deficiências.

2.2.4 Teste de Sistema

A última bateria de teste é realizada com o software interagindo com outros elementos do sistema (hardware, rede, informação, pessoas). Este teste serve para mostrar que o software está preparado para ser utilizado no ambiente para o qual foi projetado.

Por Pressman (2006), teste de sistema é uma série de diferentes testes cuja finalidade principal é testar por completo um sistema baseado em computador. Apesar de cada teste ter uma finalidade distinta, eles trabalham em conjunto para verificar se os elementos do sistema foram adequadamente integrados e executam as funções a eles alocadas.

Existem inúmeros tipos de testes de sistema; a seguir são descritos os tipos ressaltados por Pressman (2006), Myers (2004) e Rios & Moreira (2006):

- **Teste de usabilidade:** verifica o nível de facilidade de uso do software pelos usuários. Deve ser efetuado principalmente em aplicações *web*, nas quais existe muita navegação entre páginas. As telas de ajuda devem ser avaliadas quanto ao seu conteúdo e clareza de linguagem, assim como as mensagens de erro;

- **Teste de carga ou estresse:** avalia a resposta de um software sob situações anormais, que podem ser: carga pesada de dados, repetição de certas ações de entrada de dados, entrada de valores numéricos grandes e número grande de usuários simultâneos;

- **Teste de segurança:** verifica se os mecanismos de seguranças inseridos no software vão protegê-lo contra acesso interno e externo não autorizado. Geralmente, o objetivo é fazer o custo da invasão se tornar mais alto do que o valor da informação que o invasor quer obter;

- **Teste de recuperação:** são inseridas falhas no sistema para verificar a sua capacidade de recuperação. Um sistema deve ser tolerante a falhas, isto é, falhas de processamento não devem interromper a função global do sistema. Se a recuperação for automática, sem intervenção humana, a reinicialização, os mecanismos de verificação e a recuperação dos dados devem ser avaliados. Se a recuperação não for realizada pelo próprio sistema, avalia-se o tempo de reparo para determinar se estão dentro dos limites aceitáveis;

- **Teste de desempenho ou performance:** visa garantir que o sistema atende aos níveis de desempenho e tempo de resposta acordados com os usuários e definidos nos requisitos. Testes de desempenho são frequentemente acoplados com testes de estresse e usualmente requerem instrumentação de hardware e de software.

2.3 Técnicas de Teste de Software

Uma etapa de um Processo de Teste de software é o Projeto do Teste. Nela, são definidas os casos de teste, que devem satisfazer aos objetivos globais e às estratégias de teste. Essa etapa, além de todo o Processo de Teste, será detalhada no Capítulo 3.

Com o teste de software, em princípio, poderia-se detectar todos os tipos de defeitos, submetendo o software a todas as possíveis entradas, o que é denominado de teste exaustivo. No entanto, esse conjunto de entradas pode ser muito grande ou de tamanho infinito, tornando inviável a atividade de teste devido ao tempo inconcebível para a mesma. Tal limitação da atividade de teste, que não nos permite afirmar que o programa esteja 100% correto, fez com que fossem definidas as técnicas de teste e os critérios pertencentes a cada uma delas, que serão detalhadas mais adiante. Assim, é possível conduzir essa atividade de maneira sistemática.

Segundo Pressman (2006), para projetar os casos de teste, há algumas técnicas de teste de software. Essas técnicas de teste de software fornecem diretrizes sistemáticas para projetar testes que verifiquem a lógica interna e as interfaces de cada componente de software e os domínios de entrada e saída do programa de modo a descobrir erros na função, no comportamento e no desempenho do programa.

A escolha da técnica de teste depende da fase de desenvolvimento em que o teste será aplicado. Uma técnica de teste direciona a escolha de critérios para geração de casos de teste que, ao serem executados, vão testar os elementos requeridos pela abordagem do teste.

Na literatura, as técnicas de testes são divididas basicamente em duas diferentes categorias: teste caixa-branca ou estrutural e teste caixa-preta ou funcional. Algumas técnicas são para certo tipo de estrutura de software, seguindo a estratégia de teste para a qual ela será usada. Pressman (2006) cita separadamente as técnicas de teste de software orientado a objetos e as técnicas de teste baseado em estados.

Ao adotar uma técnica de teste (Funcional ou Estrutural) é necessário escolher um critério para a elaboração dos casos de teste com a finalidade de testar os elementos do

software. A seguir, são detalhados estes dois grupos de técnicas de teste, além das técnicas específicas para o paradigma orientado a objeto.

2.3.1 Teste Estrutural

Principalmente utilizado durante o teste de unidade do sistema, o teste estrutural ou caixa-branca visa testar o código propriamente dito, isto é, estabelece os casos de teste com base em uma dada implementação, executando partes ou componentes elementares do programa.

Segundo Pressman (2006), com esse tipo de técnicas, o engenheiro de software pode derivar casos de teste que:

- Garantam que todos os caminhos independentes de um módulo tenham sido testados pelo menos uma vez;
- Testem todas as decisões lógicas em seus lados verdadeiro e falso;
- Executem todos os ciclos nos seus limites e dentro de seus intervalos operacionais;
- Testem as estruturas de dados internas para garantir sua validade.

De acordo com Delamaro *et al.* (2007), é importante ressaltar limitações inerentes à técnica estrutural, como:

- **Caminhos ausentes:** se o programa não implementa algumas funções, não existirá um caminho que corresponda àquela função; conseqüentemente, nenhum dado de teste será requerido para exercitá-la;
- **Correção coincidente:** o programa pode apresentar, coincidentemente, um resultado correto para um dado particular de entrada, satisfazendo um requisito de teste e não revelando a presença de um defeito; entretanto, se escolhido outro dado de entrada, o resultado obtido poderia ser incorreto. Essa é uma limitação fundamental para qualquer estratégia de teste.

Existem variadas técnicas de teste caixa-branca citadas pelos autores (Delamaro *et al.*, 2007; Pressman, 2006; Sommerville, 2003). São exemplificadas algumas delas, sem entrar em detalhes:

- **Teste de caminho básico:** é uma das técnicas que utiliza informações sobre a complexidade do programa para derivar os casos de teste. Esta técnica garante que cada comando do programa seja executado pelo menos uma vez durante o teste;

- **Teste de condição:** exercita as condições lógicas contidas em um módulo do programa. Uma condição simples é uma variável booleana ou uma expressão relacional. Uma condição composta é formada por duas ou mais condições simples;

- **Teste de fluxo de dados:** o número de caminhos do fluxo de dados de um software pode ser grande ou infinito; por isso, surgiu a técnica de fluxo de dados, pois, a partir da análise do fluxo de dados, são derivados os casos de teste.

2.3.2 Teste Funcional

Teste funcional ou caixa-preta é uma técnica de teste utilizada após o teste estrutural, de maneira complementar, e visa verificar a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código e da lógica interna do componente testado.

A vantagem de não levar em consideração qualquer parte implementada é este tipo de técnica de teste poder ser aplicada em todas as fases de teste e em produtos desenvolvidos com qualquer paradigma de programação.

A técnica de teste funcional baseia-se apenas na especificação dos requisitos do software. Logo, para garantir a qualidade do teste, é fundamental que tais especificações sejam feitas de maneira precisa. Especificações ausentes ou incompletas tornam difícil a aplicação dos critérios funcionais.

De acordo com Delamaro *et al.* (2007), neste tipo de técnica, projetam-se casos de teste em que o programa ou sistema é considerado uma caixa preta e, para testá-lo, são fornecidas entradas e avaliadas as saídas geradas de modo a verificar se estão em conformidade com os objetivos especificados.

Pressman (2006) cita que o teste funcional tenta encontrar erros das seguintes categorias: funções incorretas ou omitidas, erros de interface, erros de estrutura de dados ou de acesso à base de dados externa, erros de comportamento e desempenho, e erros de iniciação e término.

Os testes são projetados para responder às seguintes questões (Pressman, 2006):

- 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?

Assim, tal como no teste estrutural, existem diversas técnicas de teste funcional citadas pelos autores (Delamaro *et al.*, 2007; Pádua, 2003; Pressman, 2006, Sommerville, 2003). Sem entrar em detalhes, são apresentadas algumas delas:

- **Particionamento de equivalência:** é uma técnica que divide o domínio de entrada em classes de equivalência. Uma vez definidas as classes de equivalência, pode-se assumir, com alguma segurança, que qualquer elemento da classe pode ser considerado um representante desta, pois eles devem se comportar de forma similar, reduzindo o domínio de entrada;

- **Análise de valor limite:** é uma técnica para seleção de casos de teste que exploram condições limites, as quais são valores que estão exatamente sobre ou imediatamente acima ou abaixo dos limites das classes de equivalência. Costuma-se utilizar em conjunto com a técnica de particionamento de equivalência, mas, em vez dos dados de testes serem escolhidos de forma aleatória, eles são selecionados de forma que o limitante de cada classe seja explorado;

- **Grafo causa-efeito:** diferentemente das duas técnicas anteriores, que não exploram combinações dos dados de entrada, a técnica grafo causa-efeito define casos de teste que exploram ambigüidades e incompletude nas especificações. Nesta técnica, define-se um grafo causa-efeito (entrada-ações) para, em seguida, convertê-lo em uma tabela de decisão de onde serão extraídos os casos de teste.

2.3.3 Teste Orientado a Objetos

Segundo Delamaro *et al.* (2007), a idéia da orientação a objetos é agrupar em uma entidade (denominada classe) os dados (atributos) e os procedimentos/funções (métodos) que realizam operações sobre os dados. Assim, os dados podem permanecer isolados (ou

encapsulados na classe) e o acesso a eles só pode ser realizado por meio de métodos definidos na classe, ocasionando o chamado ocultamento de informação.

Por causa dessas características únicas da orientação a objeto, alguns erros não comuns em uma programação procedimental podem ocorrer com mais freqüência em software construído usando a programação orientada a objeto. Neste tipo de programação, por exemplo, têm-se muitos métodos e, conseqüentemente, muitas interfaces, podendo aumentar a ocorrência de erros na programação de interface.

O contrário também pode ocorrer, isto é, a programação orientada a objeto pode reduzir a ocorrência de alguns tipos de erros cometidos na programação procedimental. Delamaro *et al.* (2007) cita que, em geral, métodos contêm somente algumas poucas linhas de código, erros de fluxos de controle são menos prováveis de ocorrer. O encapsulamento[3] previne defeitos resultantes do acesso a dados armazenados em variáveis globais, que podem vir a ocorrer em linguagens procedimentais.

Com isso, as técnicas e os critérios de teste para programação orientada a objetos são taticamente um pouco diferentes da programação procedimental, no entanto a estratégia é similar.

Pressman (2006) pondera que, como modelos de análise e projeto orientado a objeto são similares na estrutura e no conteúdo para o programa orientado a objeto resultante, o teste pode começar com a revisão desses modelos. Uma vez gerado o código, a atividade de teste exercita operações de classes e examina se existem erros quando uma classe colabora com outras classes. A medida que as classes são integradas para formar um subsistema, o teste baseado no uso, em conjunto com abordagens tolerantes a falhas, é aplicado para testar completamente a classes que colaboram entre si. Finalmente, casos de uso são usados para descobrir erros no nível de validação do software.

As técnicas caixa-branca e caixa-preta podem ser usadas para testar programas orientados a objetos.

O teste caixa-branca pode ser aplicado a operações definidas para uma classe. As técnicas estruturais podem ajudar a garantir que cada declaração em uma operação tenha sido testada. Porém, a estrutura concisa de muitas operações faz alguns argumentarem que o esforço aplicado ao teste caixa-branca poderia ser redirecionado para teste de nível de classe.

3 O encapsulamento se refere ao mecanismo de controle de acesso que determina a visibilidade de atributos e métodos dentro de uma classe. Com o controle de acesso, previnem-se dependências indesejadas entre uma classe cliente e uma classe servidora, por exemplo, tornando visível ao cliente somente a interface da classe, ocultando detalhes de implementação (Delamaro *et al.* 2007).

As técnicas de teste caixa-preta também são adequadas para sistemas orientado a objetos, visto que essas técnicas não levam em consideração o tipo de programação usado para o desenvolvimento dos sistemas.

2.4 Considerações Finais

No presente capítulo, foi apresentada a importância da atividade de teste no atual processo de desenvolvimento de software. Diferentemente do que se pensa, o teste não deve ser conduzido apenas no final do processo de desenvolvimento, ele deve ser realizado ao longo do processo, para que erros sejam encontrados a partir do início do desenvolvimento e, com isso, economizar custo e tempo na solução deles.

Um bom teste não é aquele que certifica que um software não contém erros. Ao contrário, o objetivo do teste é encontrar o maior número de erros antes que o software seja entregue ao cliente.

A atividade de teste, atualmente, é feita de forma organizada, existem estratégias, métodos e técnicas de teste que ajudam a tornar a atividade mais eficaz, apesar de muitos conceitos serem novidades, o que torna um pouco difícil o entendimento. Essas estratégia, métodos e técnicas compõem o Processo de Teste, tema do próximo capítulo.

3 PROCESSO DE TESTE

No mundo atual, existe uma demanda não satisfeita por software de qualidade. Empresas de desenvolvimento de software sofrem cada vez mais pressão para construir sistemas de informação em um espaço de tempo mínimo. Tais sistemas precisam ser escalonáveis e devem interagir com outros sistemas existentes ou em desenvolvimento.

O emprego de sistemas de informação nos mais diversos setores da sociedade cresce ao ponto em que boa parte dos negócios depende de software e computadores. Esta dependência exige que software produzido seja de qualidade.

Pressman (2006) recorda que o American Heritage Dictionary define qualidade como "característica ou atributo de alguma coisa". Como atributo de um item, a qualidade se refere a características mensuráveis – coisas que podem ser comparadas com padrões conhecidos tais como comprimento, cor, propriedades elétricas e maleabilidade. No entanto, é mais difícil de caracterizar o software que objetos físicos, por ele ser essencialmente uma entidade intelectual. Mas hoje em dia, já existem medidas das características de um programa. Essas propriedades incluem a complexidade ciclomática, a coesão, o número de pontos por função e de linhas de código, entre muitas outras.

Para o desenvolvimento de software com qualidade, dentro de prazos e custos controlados e compatíveis com mercado, é fundamental a melhoria de processos da engenharia de software. Abordagens e experiências para a melhoria de processo de software baseada em modelos têm sido utilizadas com sucesso pelas empresas. Os modelos mais comuns são: ISO/IEC[4] 12119, ISO/IEC 12207, ISO/IEC 15504, CMMI[5].

Estes modelos têm a função de melhorar os processos de desenvolvimento de software e, nestes modelos, o teste de software é citado direta ou indiretamente. Teste é fundamental para a avaliação do software desenvolvido, sendo uma forma de agregar qualidade ao software e dando maior confiabilidade a ele.

Antes de entrar em detalhe sobre processo de teste, é interessante tratar o assunto qualidade de software, pois o teste pode ser considerado um atestado para tal.

4 International Organization for Standardization/International Electrotechnical Commission
5 Capability Maturity Model Integration

3.1 Qualidade de Software

Como citado anteriormente, cada vez mais os clientes demandam software com qualidade. No entanto, Pressman (2006) argumenta que não é suficiente dizer que qualidade de software é importante, mas:

- definir explicitamente o que é "qualidade de software";
- criar um conjunto de atividades que ajudarão a garantir que todo o produto do trabalho de engenharia de software exiba alta qualidade;
- realizar atividades de controle e de garantia de qualidade de software no projeto;
- usar métricas para desenvolver estratégias para aperfeiçoar seu processo de software e, como conseqüência, a qualidade do produto final.

Pressman (2006) ainda conceitua qualidade de software como a satisfação de requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas esperadas em software desenvolvido profissionalmente.

Este conceito atende à visão do usuário sobre um software de qualidade, pois os usuários estão mais interessados no uso do software, no seu desempenho e nos efeitos que o seu uso possa produzir na organização. Eles não valorizam os aspectos internos do software ou como ele foi desenvolvido. No entanto, é possível chegar a essa qualidade externa, exigida pelo cliente, observando a qualidade interna da organização.

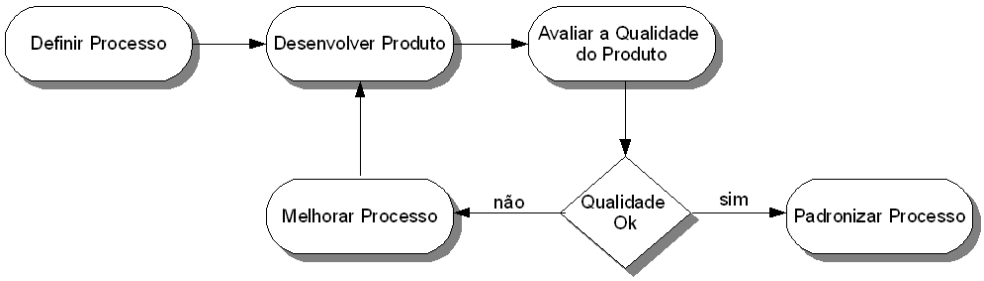

Figura 3.1: Qualidade Baseada em Processos (Fonte: Sommerville, 2003)

Uma suposição básica do gerenciamento da qualidade é a qualidade do processo de desenvolvimento definido afetar diretamente a qualidade dos produtos fornecidos. A Figura 3.1 ilustra essa suposição. O processo primeiramente é definido, para que na seqüência o produto seja desenvolvido seguindo as diretrizes do processo. Se o produto final for de qualidade, o processo pode ser padronizado; caso contrário, deve haver a melhora do processo.Mas, como se consegue um controle de qualidade? Controle de Qualidade envolve uma série de inspeções,

revisões e testes usadas ao longo do processo de software para garantir que cada produto de trabalho satisfaça os requisitos para ele estabelecidos.

O Controle de Qualidade supervisiona o processo de desenvolvimento de software para assegurar que sejam seguidos os procedimentos e os padrões de garantia da qualidade do software.

3.1.1 Revisões

As revisões são grandes aliadas no Controle de Qualidade de Software. Pressman (2006) afirma que as revisões são um "filtro" para o processo de engenharia de software. Elas são aplicadas em vários pontos durante a engenharia de software, a fim de encontrar erros durante o processo, de modo que eles não se transformem em defeitos após a entrega do software.

Não é definido de quem é a função de executá-las. Pádua (2003) sugere que fique sob responsabilidade do grupo independente de Garantia da Qualidade do Software (citada na seção 3.1.2).

Uma variedade de revisões podem ser conduzidas como parte da engenharia de software. Na Tabela 3.1, são descritos de forma superficial alguns tipos de revisão.

Tabela 3.1: Tipos de Revisão (Fonte: Sommerville, 2003)

Tipos de Revisão	Objetivo Principal
Inspeção de Projeto ou Programa	Detectar erros 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 do 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.

3.1.2 Garantia da Qualidade de Software

O propósito da Garantia da Qualidade de Software é fornecer à gerência a visibilidade da eficácia do processo utilizado pelo projeto de desenvolvimento de software e da qualidade dos artefatos criados. Ela é o exame minucioso de um estado do projeto ou de um artefato de

software, com a finalidade de determinar se há algum desvio com relação aos padrões, diretrizes, especificações, procedimento e planos aprovados e para recomendar melhorias.

Para Pádua (2003), os objetivos da Garantia da Qualidade de Software são:

- dar suporte ao trabalho das equipes de software para garantir a qualidade de atividades e resultados de cada processo;
- verificar as atividades e os resultados dos projetos de software;
- fornecer à gerência informação atualizada sobre o status dos projetos e sua conformidade com os padrões de qualidade da organização.

Normalmente, a Garantia da Qualidade de Software é responsabilidade de um grupo formado por profissionais da engenharia de software, independente do projeto e da gerência do projeto. O trabalho é realizado em conjunto com a equipe do projeto desde o seu início.

Pádua (2003) cita alguns outros possíveis responsáveis pela garantia da qualidade de software:

- os próprios desenvolvedores e seus pares, através do uso fiel dos processos e das revisões formais e informais;
- os gerentes de projeto, embora limitados por não terem a visão independente de um grupo externo ao projeto;
- os grupos externos de verificação e validação independentes, que podem ser chamados para servir de mais um nível de controle, em projetos de alto risco, ou por exigência do cliente.

Se os dados fornecidos por meio da garantia da qualidade identificam problemas, é responsabilidade da gerência cuidar dos problemas e aplicar os recursos necessários para resolver as questões de qualidade.

A Tabela 3.2 identifica as principais atividades de Garantia de Qualidade:

Tabela 3.2: Atividade de Garantia de Qualidade (Fonte: Pádua, 2003)

Atividade	Descrição sucinta	Responsável	Insumos	Resultados
Planejamento	Confecção das partes referentes à Garantia da Qualidade do Plano da Qualidade do Software	Gerente de Projeto	Plano de Desenvolvimento de Software	Plano da Qualidade do Software
Auditoria da Qualidade	Verificação da conformidade com o processo dos artefatos e atividades de um projeto	Grupo de Gestão de Qualidade de Software	Plano de Desenvolvimento de Software; Plano da Qualidade do Software; artefatos	Relatório de Auditoria da Qualidade do Software; solicitações de providências
Verificação da Organização	Verificação da conformidade com o processo das atividades da organização, não vinculadas a projetos	Grupo de Gestão de Qualidade de Software	Estrutura da Organização	Relatórios da Organização

Na atividade de Planejamento, o executor é o gerente de um projeto específico. O principal insumo é o Plano de Desenvolvimento do Software e o resultado dessa atividade é o Plano de Qualidade do Software.

Na atividade de Auditoria da Qualidade, cabe ao grupo independente de Garantia da Qualidade executá-la. O grupo recebe do projeto os documentos e os itens a serem verificados, os marcos de entrega e o conteúdo desses documentos. As discrepâncias encontradas dão origem a solicitações de providências, encaminhadas ao gerente de projeto. Se o grupo e o gerente não encontrarem soluções, a questão é encaminhada à gerência executiva. Os resultados de cada auditoria gera um Relatório da Audiência da Qualidade do Software.

Uma outra atividade realizada pelo grupo independente de Garantia de Qualidade é a Verificação da Organização. Nela, são verificadas as atividades de outros grupos não ligados a projetos específicos, como os de Gerência de Configurações e de Engenharia de Processos. O resultado gerado é relatórios a serem encaminhados à gerência executiva.

3.2 Qualidade no Processo de Teste

Segundo Sommerville (2003), o teste de software tem o objetivo de garantir que o produto final atenda aos requisitos desejados e que seja capaz de tratar corretamente os tipos possíveis de dados de entrada, contribuindo para a Garantia da Qualidade de Software.

Como mencionado anteriormente, os modelos utilizados na melhoria do processo de desenvolvimento de software das organizações citam o teste como atividade fundamental para agregar qualidade ao software.

O processo proposto neste trabalho será baseado em duas normas: na norma brasileira ISO/IEC 12119 (NBR ISO/IEC 12119) que estabelece os requisitos de qualidade para pacotes de software e instruções para teste, considerando esses requisitos, e na norma *Institute of Electrical and Electronics Engineer 829* (IEEE 829), que descreve um conjunto de documentos para as atividades de teste de um produto de software.

Foi estudada a possibilidade de se utilizar a Norma ISO/IEC 15504-5. Ela é uma evolução da ISO/IEC 12207 e o seu foco é a avaliação para melhoria do processo de desenvolvimento de software com níveis de capacidade para cada processo. Porém, ela trata processos já definidos e implantados em uma organização e, no presente trabalho, essa não é a realidade encontrada. Nas próximas seções, é detalhada cada uma destas duas normas utilizadas para a realização deste trabalho.

3.3 NBR ISO/IEC 12119

A NBR ISO/IEC 12119 é equivalente à ISO/IEC 12119-1994 e foi validada pela Associação Brasileira de Normas Técnicas (ABNT) em 1998. Ela é aplicável a pacotes de software, na forma como eles são oferecidos e liberados para uso, mas não trata processos de produção de software.

Esta norma estabelece dois itens que devem ser validados em um pacote de software: requisitos de qualidade para pacote de software e instruções de como testar um pacote de software com relação aos requisitos estabelecidos.

3.3.1 Requisitos de Qualidade

A NBR ISO/IEC 12119 indica a necessidade de que cada pacote de software tenha uma descrição do produto e uma documentação para o usuário. Neste item, são estabelecidos requisitos para tal descrição e documentação, além de requisitos para programas e dados inclusos no pacote.

A conformidade de um produto em relação aos requisitos definidos pode ser difícil ou impossível de provar, mas um teste (incluindo revisão de documentação) de acordo com um

requisito é considerado suficiente para proporcionar a confiança necessária para certificar que o produto está em conformidade com a norma.

Os requisitos de qualidade estabelecidos nesse item são para: (i) Descrição do Produto; (ii) Documentação do Usuário; (iii) Programas e Dados.

(i) Descrição do Produto

A descrição define o produto e faz parte do seu conjunto de documentação. Ela auxilia o usuário ou o comprador na avaliação da adequação do produto às suas necessidades, além de poder fornecer informações para vendas. Além disso, serve como base para testes.

Os requisitos que devem estar contidos na descrição do produto são:

- **Requisitos gerais sobre o conteúdo da descrição:** convém ser suficientemente inteligível e completo, e ter boa organização e apresentação; deve ser livre de inconsistências internas; cada declaração da descrição de produto deve ser correta e passível de teste;

- **Identificações e indicações:** devem conter identificações como da descrição, do produto e dos componentes físicos (documentos e meios de armazenamento de dados); também devem conter indicações sobre fornecedor, tarefas, requisitos de hardware e software, instalação, suporte e manutenção;

- **Declarações sobre funcionalidade:** devem incluir uma visão geral sobre a funcionalidade do produto, os dados necessários e as facilidades oferecidas, os valores-limites, caso o produto seja limitado por estes, e maneiras de evitar acesso não autorizado, caso seja necessário;

- **Declarações sobre confiabilidade:** devem incluir informações sobre procedimentos para preservação de dados, isto é, os procedimentos devem ser empregados para assegurar a capacidade funcional do produto;

- **Declarações sobre usabilidade:** devem conter requisitos para melhor uso do produto, como o tipo de interface utilizada, o conhecimento necessário para manuseio, os idiomas utilizados no produto e na documentação, adaptação às necessidades do usuário, proteção contra infrações a direitos autorais;

- **Declarações sobre eficiência:** podem incluir dados sobre o comportamento do produto em relação ao tempo de resposta;

- **Declarações sobre manutenibilidade:** podem conter declarações sobre manutenibilidade;
- **Declarações sobre portabilidade:** podem conter declarações sobre portabilidade.

Há de enfatizar que alguns requisitos estabelecidos são mandatários, isto é, devem ser inclusos, outros são recomendados, sendo utilizados de acordo com o produto.

(ii) Documentação de Usuário

São requisitos estabelecidos para a documentação utilizada pelo usuário ou por quem for utilizar o produto. Foram estabelecidos os seguintes requisitos:

- **Completitude:** devem ser completamente descritas as funções estabelecidas na descrição do produto e as funções do programa que os usuários têm acesso;
- **Correção:** as informações devem estar corretas e não ambíguas;
- **Consistência:** não deve apresentar contradições internas entre si e com a descrição do produto;
- **Inteligibilidade:** convém que seja inteligível para os usuários que normalmente executam as tarefas atendidas pelo produto;
- **Apresentação e organização:** convém possuir boa apresentação e organização. Se não estiver na forma impressa, convém que seja indicado o procedimento para impressão.

(iii) Programas e Dados

São estabelecidos os requisitos para programa e dados adicionais ao pacote de software, caso eles existam. A norma estabelece os seguintes requisitos:

- **Funcionalidade:** deve ser possível instalar os programas com sucesso seguindo o manual de instalação, caso seja realizada pelo usuário, respeitando os requisitos de hardware e software contidos na descrição do produto; as funções devem ser executadas na forma descrita; programas e dados não devem conter contradições internas com a documentação e com a descrição do produto;
- **Confiabilidade:** o sistema, incluindo hardware e software, deve estar de tal modo que o usuário consiga controlá-lo, não devendo corromper ou perder dados, mesmo quando o usuário o expuser a limites máximos ou a entradas inválidas;

- **Usabilidade:** é conveniente que os programas tenha mensagens, perguntas e resultados inteligíveis, boa organização e apresentação, de modo que o usuário tenha informações claras e fáceis de serem lidas e advertências claras sobre execuções que podem trazer conseqüências graves, como apagar dados;

- **Eficiência, manutenibilidade e portabilidade:** o produto deve estar em conformidade com as declarações de eficiência, manutenibilidade e portabilidade, citadas em sua descrição, porém não sendo exigido.

3.3.2 Instruções para Teste

Nesta seção, são especificadas instruções para testar um produto em relação aos requisitos de qualidade, incluindo o teste das propriedades necessárias aos produtos de mesmo tipo e o teste das propriedades especificadas na descrição do produto. Além disso, é incluído teste por inspeção de documentos (revisão).

Estas instruções contêm a descrição do teste funcional (caixa-preta). O teste estrutural (caixa-branca) não está incluso por requerer a disponibilidade do código fonte. No entanto, durante a produção, pode ser mais barato ou mais eficaz usar o teste estrutural.

São estabelecidos nas instruções os seguintes passos: (i) Pré-requisitos de Teste; (ii) Atividades de Teste; (iii) Registros de Teste, (iv) Relatórios de Teste e (v) Teste de Acompanhamento.

(i) Pré-requisitos de Teste

São descritos os pré-requisitos necessários para fazer os testes no pacote de software. Os pré-requisitos são:

- **Presença de itens do produto:** devem estar presentes itens entregues, como documentação impressa e meios de armazenamento de dados;

- **Presença de componentes do sistema:** devem estar disponíveis as partes constituintes do ambiente de hardware e de software, identificadas na descrição de produto;

- **Treinamento:** se o treinamento for mencionado na descrição do produto, o testador deverá ter acesso ao material e ao programa de treinamento.

(ii) Atividades de Teste

A descrição do produto, a documentação e os programas e dados, que constituem o pacote de software, devem ser verificados quanto ao cumprimento dos requisitos de qualidade, estabelecidos anteriormente e é importante que sejam verificados quanto ao cumprimento das recomendações citadas.

Com relação aos programas, eles devem ser testados nos ambientes de hardware e de software, especificados na descrição do produto. Além disso, programas e dados devem ser testado usando guias de teste (casos de teste), construídos com base na descrição de produto e na documentação do usuário.

Os objetivos do teste são derivados desses requisitos. Os detalhes não testados devem constar nos registros e relatórios de teste, assim como o motivo de não testá-los.

(iii) Registros de Teste

Os registros devem conter informações suficientes para repetir o teste. Portanto, devem ser incluído um plano de teste contendo os casos de teste, os resultados destes casos, incluindo falhas ocorridas durante o teste, bem como a identificação do pessoal envolvido.

(iv) Relatório de Teste

No relatório, devem constar os objetivos e os resultados do teste, possuindo a seguinte estrutura: identificação do produto; sistemas computacionais usados no teste; documentos usados; resultados de teste da descrição de produto, documentação do usuário, programas e dados; listas das não-conformidades com os requisitos mandatários e com os recomendados; e data de encerramento do teste.

(v) Teste de Acompanhamento

A norma estabelece que, quando um produto for testado novamente, as partes modificadas nos documentos, funções e dados, bem como as partes inalteradas mas possíveis de serem influenciadas pelas partes modificadas devem ser testadas como se fossem um produto novo. As outras partes também devem ser testadas utilizando casos de teste com seleção por amostragem.

Enfim, foi visto até aqui um resumo da NBR ISO/IEC 12119. A próxima seção é um resumo da Norma IEEE 829, que estabelece documentos para o processo de teste.

3.4 Norma IEEE 829

A norma IEEE Std 829-1998 é uma revisão da IEEE Std 829-1983 e descreve um conjunto básico de documentos para acompanhamento e controle do processo de teste de software, especificando a forma e o conteúdo de cada um destes documentos.

Ela define oito documentos usados durante as fases do processo de teste. O documento (i) Plano de Teste cobre a fase de planejamento. A fase de especificação é coberta pelos documentos (ii) Especificação de Projeto de Teste, (iii) Especificação de Caso de Teste, (iv) Especificação de Procedimento de Teste. Os documentos (v) Relatório de Encaminhamento de Itens de Teste, (vi) Relatório de Teste, (vii) Relatório de Incidente de Teste e (viii) Relatório de Resumo de Teste cobrem a fase de relatórios de teste (VILLAS BOAS, 2003).

A seguir, serão apresentados tais documentos, de forma resumida.

(i) Plano de Teste

Descreve o escopo, os métodos, os recursos e o cronograma das atividades de teste. Identifica os itens e as características a serem testados, as tarefas de teste a serem executadas, o pessoal que executará cada tarefa e os riscos associados ao plano.

(ii) Especificação de Projeto de Teste

Especifica, de maneira detalhada, os métodos de teste e identifica as características a serem testadas por esse projeto, e os testes associados a ele. Além disso, identifica e descreve brevemente os casos de teste e os procedimentos (caso existam) e especifica os critérios de aceitação utilizados para aprovar ou não uma determinada característica.

(iii) Especificação de Caso de Teste

Especifica, de maneira detalhada, um caso de teste definido no projeto de teste. Descreve os itens a serem testados, as entradas necessárias, as saídas esperadas, o ambiente necessário, os procedimentos especiais e as dependências a outros casos de teste, caso existam.

(iv) Especificação de Procedimento de Teste

Especifica os passos necessários para executar um conjunto de casos de teste.

(v) Relatório de Encaminhamento de Itens de Teste

Identifica os itens a serem encaminhados para testes, incluindo a pessoa responsável pelos itens e sua localização.

(vi) Registro de Teste

Fornece um registro cronológico de detalhes relevantes à execução dos testes, indicando qual procedimento é executado e relaciona (caso haja) os relatórios de incidentes dos testes realizados.

(vii) Relatório de Incidentes de Teste

Documenta qualquer evento ocorrido no processo de teste e que mereça ser analisado, indicando os itens de teste envolvidos, o procedimento de teste executado, os casos de teste utilizados e o registro de teste relacionado.

(viii) Relatório de Resumo de Teste

Resume os resultados das atividades de teste desenvolvidas e fornece uma avaliação baseadas nesses resultados. Deve descrever todos os itens testados e toda a documentação relativa ao teste.

A Figura 3.2 mostra o relacionamento entre os documentos de teste.

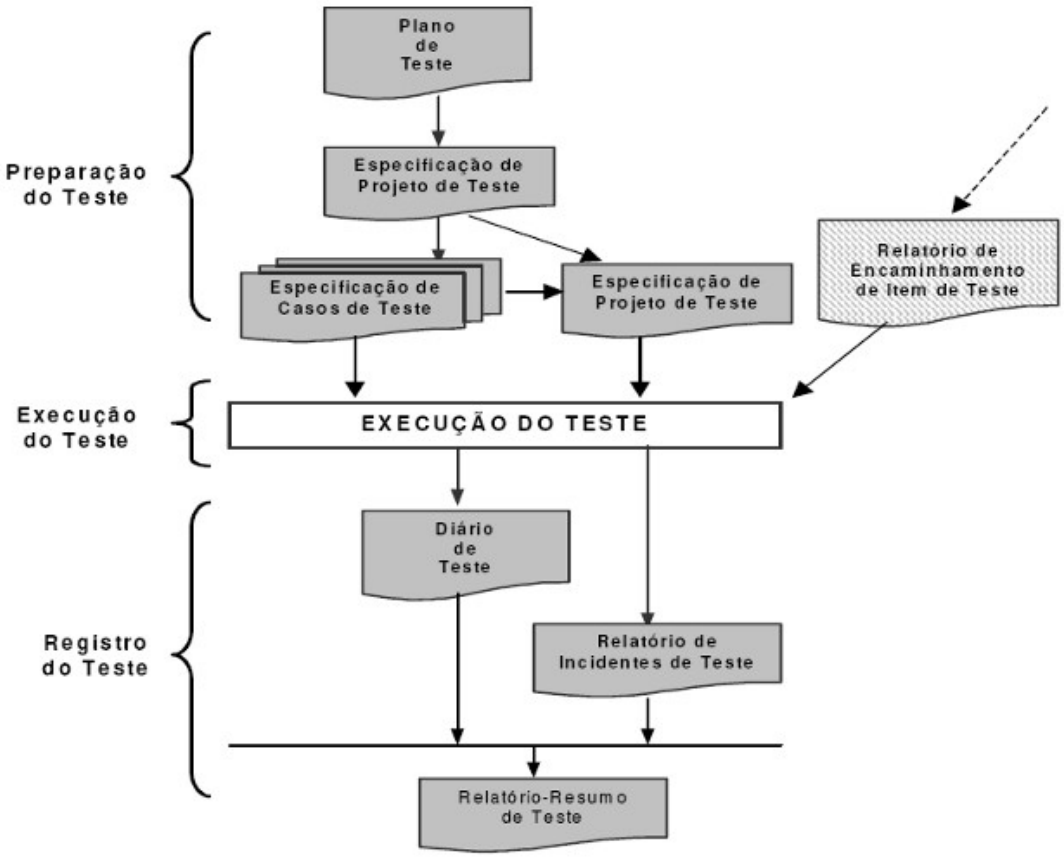

Figura 3.2: Relacionamento entre os Documentos de Teste (Fonte: Crespo 2004)

3.5 Etapas do Processo de Teste

O processo de teste é caracterizado por um conjunto de atividades de teste executadas ao longo do ciclo de desenvolvimento do software. Essas atividades estão agrupadas em etapas bem definidas, citadas por Villas Boas (2003):

1. **Planejamento:** são distribuídos, de forma racional no tempo, os recursos disponíveis para a realização da atividade de teste, decidindo antecipadamente o que deve ser feito e quando deve ser feito. Nessa etapa, é descrito o escopo dos testes, identificando os métodos empregados, os recursos necessários, o cronograma de atividades, o pessoal necessário, os itens a serem testados, os itens a não serem testados, as características dos itens testados e as responsabilidades. Como resultado dessa etapa, tem-se o Plano de Teste;

2. **Projeto:** é especificada de maneira detalhada a forma como será realizada a atividade de teste. Nessa etapa, é escolhido um grupo específico de características a serem testadas, descrevendo detalhadamente os métodos empregados e os testes que deverão ser feitos. O resultado dessa etapa é o documento Projeto de Teste;

3. **Implementação:** são fornecidos subsídios indispensáveis à execução de cada tarefa de teste. Nessa etapa é descrito detalhadamente os itens a serem testados, as entradas utilizadas, as saídas esperadas e os passos necessários para executar os testes, ou seja, a seqüência de tarefas necessárias para analisar um item de software com a finalidade de avaliar um conjunto de suas características. Como resultado, têm-se o Caso de Teste e o Procedimento de Teste;

4. **Execução:** É a parte prática do teste, isto é, aquela em que se efetiva a tarefa de teste propriamente dita. O testador realiza os testes a partir do Procedimento de Teste e dos Casos de Teste associados. Nesta etapa, são registradas todas as atividades de teste, identificando a data e a hora do teste, o autor do teste, o procedimento seguido, o pessoal envolvido, os resultados obtidos, as condições ambientas e os eventos não esperados (caso ocorram). Caso aconteçam problemas durante o teste, estes deverão ser relatados, sendo referidos o procedimento em uso, os casos de teste associados e os itens testados, fornecendo uma descrição do problema, 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, um resumo deve ser fornecido referindo-se aos itens testados, sumarizando os resultados obtidos e fornecendo uma avaliação baseada

nesses resultados. Ao fim da etapa, são gerados os documentos Registro de Teste, Relatório de Problemas de Teste e Relatório de Resumo de Teste.

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.

Na fase de projeto lógico, após a fase de Especificação de Requisitos, deve ser gerado um Plano de Teste de Sistema, a partir do documento em que os requisitos do sistema estão especificados. Com base nesse Plano, são gerados os Projetos de Teste e os Casos de Teste do Sistema.

Na fase de projeto físico, baseado nas especificação das interfaces, devem ser gerados o Plano, o Projeto e os Casos de Teste de Integração.

Na fase de implementação, são gerados os documentos Procedimentos de Teste, que refere-se às estratégias de teste (unidade, integração, sistema). O programador executa o teste de unidade em cada implementação sob sua responsabilidade. Caso encontre o problema, ele o corrige e executa novamente o teste. Quando um grupo de unidades que compõem um módulo passa pelo teste de unidade, o teste de integração é realizado por quem foi delegado. Se encontrar problema, é feito o mesmo procedimento do teste de unidade. Se os módulos passarem pelo teste de integração, pode ser executado o teste de sistema. Havendo problemas, mais uma vez o defeito deve ser corrigido e o teste de sistema deve ser executado novamente.

3.6 Implantação de um Processo de Teste

De acordo com Rios & Moreira (2006), pode-se citar os procedimentos a serem seguidos para implantar um Processo de Teste, mesmo que pequeno ou em caráter experimental, e/ou capacitar uma equipe de teste para ajudar na garantia de qualidade das aplicações a serem desenvolvidas na organização. Basicamente, os procedimentos a adotar são: (i) Escolher a Metodologia de Testes; (ii) Escolher a Equipe de Testes; (iii) Preparar o Ambiente; (iv) Definir a Estrutura Organizacional; (v) Gerenciar o Processo de Teste; (vi) Melhorar o Processo de Teste; (vii) Automatizar o Processo de Teste. Cada um destes procedimentos são detalhados a seguir.

(i) Escolher a Metodologia de Testes

Primeiramente, deve-se escolher a metodologia de testes adequada à metodologia de desenvolvimento da organização. Significa definir o processo de teste, suas etapas, as estratégias e as técnicas a serem utilizadas. A definição do processo deve ser discutida

juntamente com a equipe de desenvolvimento, para haver melhor integração entre o processo de desenvolvimento de software e o processo de teste de software.

O uso dos artefatos baseados na IEEE 829 deve ser discutido para que sejam feitas adequações necessárias e, assim, sejam utilizados da melhor maneira no processo de teste da organização.

(ii) Escolher a Equipe de Testes

Após a metodologia ser escolhida e aprovada, o passo seguinte é criar a equipe de teste e capacitá-la. A Tabela 3.3 mostra o perfil dos profissionais que compõem uma equipe de teste.

Tabela 3.3: Perfil de um Profissional de Teste (Fonte: Rios & Moreira, 2006)

Profissional	Perfil
Líder do Projeto de Teste	responsável pela liderança de um projeto de teste específico, normalmente relacionado a um sistema de desenvolvimento
Arquiteto de Teste	responsável pela montagem da infraestrutura de teste, montando o ambiente e escolhendo ferramentas
Analista de Teste	responsável pela modelagem e elaboração dos casos de teste e dos *scripts*
Testador	responsável pela execução dos casos de teste e dos scripts

Dependendo do tamanho da equipe e da organização, um único profissional pode desempenhar mais de uma destas funções.

(iii) Preparar o Ambiente

Um esboço dos três ambientes necessários (desenvolvimento, teste e produção) pode ser visto na Figura 3.3.

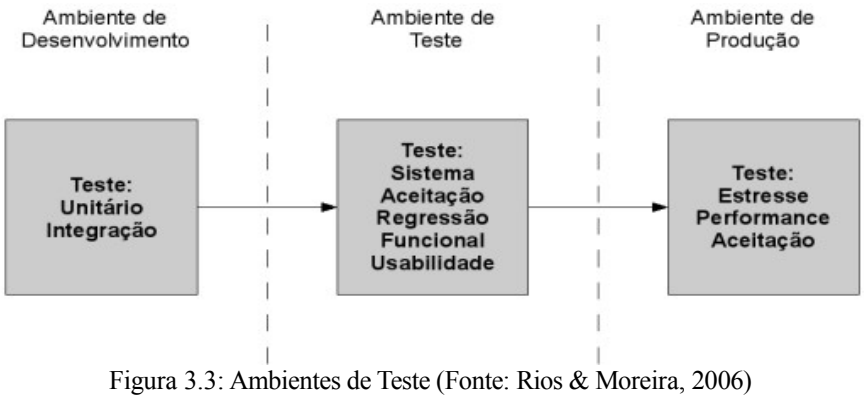

Figura 3.3: Ambientes de Teste (Fonte: Rios & Moreira, 2006)

Na Figura 3.3, estão indicados os tipos básicos de testes a serem executados em cada ambiente, embora existam outros tipos de testes, como mencionado no Capítulo 2. O ideal é o ambiente de teste estar o mais próximo possível, em termos de software e hardware, do ambiente de produção.

(iv) Definir a Estrutura Organizacional

A estrutura organizacional deve ser definida de acordo com as características de cada organização, portanto a estrutura de teste será adequada à estrutura organizacional existente, de forma que a implantação dos novos papéis traga o menor impacto possível.

(v) Gerenciar o Processo de Teste

Após ser definido e implantado o processo de teste, deve-se gerenciá-lo através de medições e indicadores, métricas para a medição do tamanho do projeto de teste e do esforço necessário a sua execução, além da administração de erros.

(vi) Melhorar o Processo de Teste

Uma vez implantado um processo de teste em um ambiente apropriado, tendo sido definido uma metodologia de teste, escolhida e capacitada uma equipe de teste, bem como o projeto sendo gerenciado de forma eficiente, o procedimento seguinte é fazer uma reavaliação, para encontrar os pontos fracos, de modo a melhorar continuamente o processo de teste implantado.

(vii) Automatizar o Processo de Teste

Uma das formas de melhoria do processo de teste é automação da atividade, porém ela deve somente ser realizada após o processo estar bem definido e a organização estar familiarizada com ele. Assim, será mais fácil escolher as melhores ferramentas de automação.

3.7 Considerações Finais

No capítulo, foi explicada a importância da Qualidade de Software para uma empresa desenvolvedora de software, visto que o termo qualidade é cada vez mais exigido pelos clientes.

Para desenvolver um software de qualidade, existem modelos de melhoria e de definição de processo de teste, os quais citam o teste como uma das ferramentas importantes para a garantia da qualidade.

A NBR ISO/IEC 12119 estabelece requisitos de qualidade e instruções de teste para serem aplicados em um pacote de software. A norma IEEE 829 define os artefatos necessários para o processo de teste.

O processo de teste pode ser definido em quatro etapas: planejamento, projeto, implementação e execução. Nelas, estão agrupadas as atividades necessárias para serem executadas ao longo do processo de desenvolvimento do software.

Implantar um processo de teste em uma organização não é uma tarefa trivial, é trabalhosa, demanda tempo, planejamento antecipado e a participação da equipe. Entretanto, uma vez definido e implantado, o processo traz proveitosos benefícios para a organização, em termos de custo, capacidade e qualidade do produto.

4 PROCESSO DE TESTE DA TECNOLIVRE

O processo de teste proposto no presente trabalho, visa atender a necessidade de uma cooperativa que presta serviços de informática, TecnoLivre – Cooperativa de Tecnologia e Soluções Livres. O cenário atual da cooperativa foi levado em consideração.

O processo de desenvolvimento de software da TecnoLivre foi proposto no trabalho de Coelho (2007). Neste processo, está inclusa a fase de teste. No entanto, essa proposta está defasada, pois, conforme os últimos estudos, citados nos capítulos anteriores, a atividade de teste deve ocorrer ao longo do processo de desenvolvimento, não apenas no fim.

Além disso, a fase de teste proposta não tem um planejamento detalhado de como serão realizados os testes e a falta de conhecimento específico dos profissionais da cooperativa diminuiu ainda mais a eficiência dessa atividade, resultando em uma execução da fase de teste fora do modelo definido no processo.

Foi realizada uma pesquisa para levantamento de processos de teste existentes para servirem de embasamento para a elaboração da proposta deste trabalho. Os estudos sobre os resultados práticos dos processos de teste são escassos, há pouca literatura sobre o assunto e os processos de teste encontrados não satisfazem por inteiro as necessidades da cooperativa. Dois modelos foram considerados para a elaboração da proposta, a metodologia de Rios & Moreira (2006) e a proposta de Silva (2006).

Após estudos gerais sobre engenharia de software, garantia de qualidade, atividade de teste e processo de testes existentes, foi definido o processo de teste da TecnoLivre, detalhado a seguir.

4.1 Concepção do Processo

A definição do processo de teste da TecnoLivre teve início a partir do problema de falta de organização desta atividade no processo de desenvolvimento de software da cooperativa.

O processo de desenvolvimento foi proposto por Coelho (2007), que buscou definir um processo para a TecnoLivre a partir do estudo de outros processos existentes. Baseou-se em quatro processos, a saber: Rational Unified Process (RUP), Processo de Desenvolvimento de Software 2 (PDS2), Processo para Aplicativos eXtensíveis InterativoS (PRAXIS) e eXtreme Programming (XP).

O resultado final é um processo simples e objetivo, que ao mesmo tempo, prioriza a documentação e a definição de cada papel nas atividades de cada fase do processo. O fluxograma do processo proposto por Coelho (2007) pode ser visto na Figura 4.1.

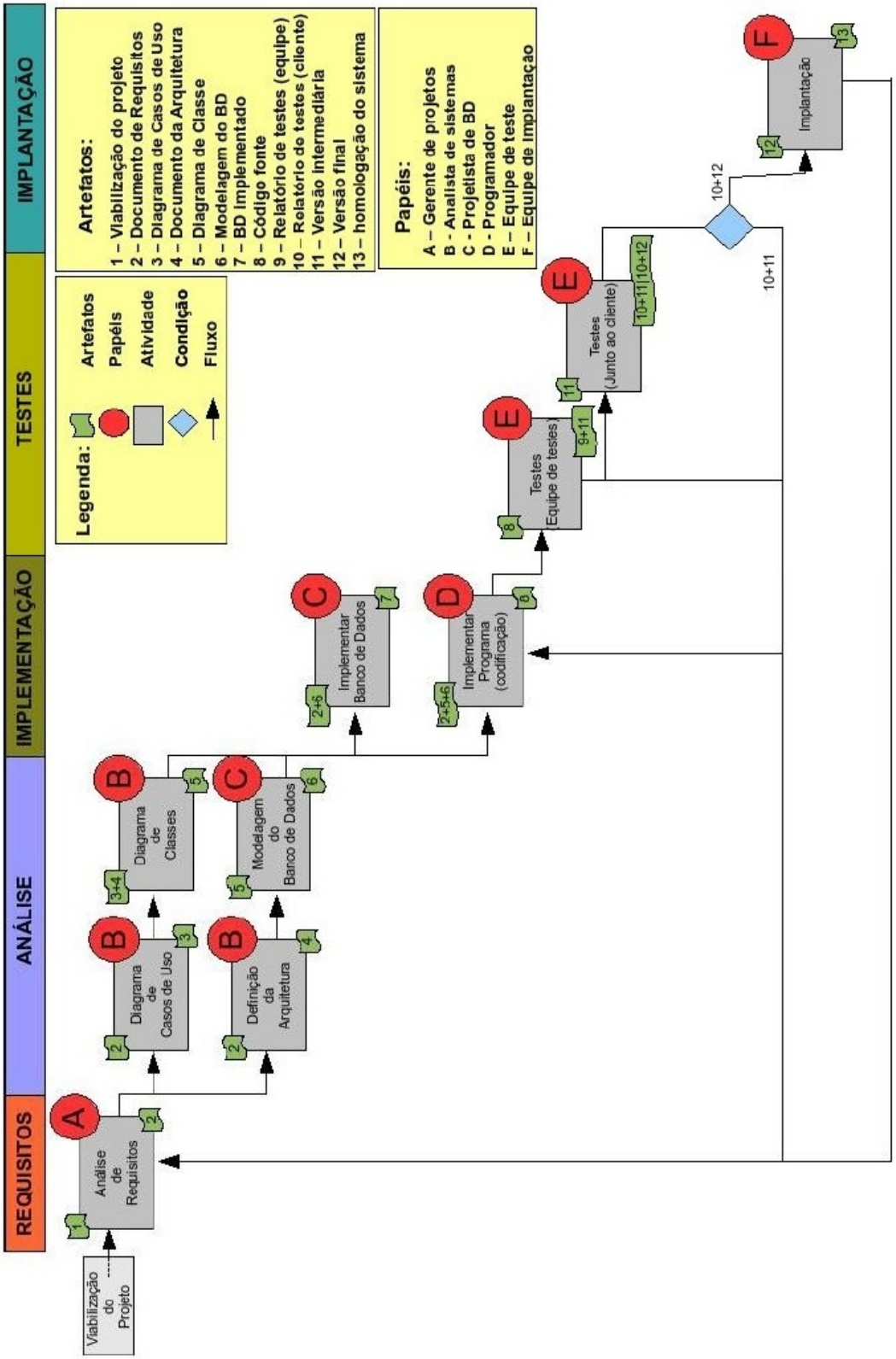

Figura 4.1: Processo de Desenvolvimento de Software (Fonte: Coelho, 2007)

Durante a fase de Requisitos do processo de desenvolvimento de software, proposto por Coelho (2007), é realizado o levantamento dos requisitos. Na fase de Análise, é realizado, além da própria análise, o projeto do software a ser desenvolvido.

Foi proposto que o teste seja realizado após a implementação e anterior à fase de implantação, sendo executado em duas etapas:

- Testes Internos;
- Testes Junto ao Cliente.

Os **testes internos** têm o intuito de verificar a conformidade da versão do software gerada na fase de implementação com os requisitos levantados. Durante esta atividade, é gerado um Relatório de Testes, que descreve os passos dos testes realizados, os resultados obtidos e a solicitação de modificações, caso sejam necessárias. Os testes são realizados no código fonte gerado, que deverá ser compilado, e assim gerar uma versão executável do software. Caso seja encontrado algum erro no relatório, é solicitado, aos responsáveis pela análise de requisitos ou pela implementação, a correção necessária.

Os **testes junto ao cliente** são realizados sobre a versão gerada na etapa anterior. É mostrada para o cliente esta versão, que aprova ou não o software desenvolvido. É gerado um outro Relatório de Testes, que também descreve os passos dos testes realizados, os resultados obtidos e a solicitação de modificações, caso sejam necessárias. Se o cliente aprovar o software desenvolvido, é gerada uma versão final a ser implantada na fase seguinte. Caso contrário, é solicitada a correção necessária aos responsáveis pela análise de requisitos ou pela implementação, através do relatório gerado.

Ambos os testes são de responsabilidade de uma Equipe de Teste, que é formada pela própria equipe participante do desenvolvimento do projeto.

Essa definição da fase de teste não satisfaz o conceito atual do teste ser realizado paralelamente ao longo do processo de desenvolvimento de software. Quanto antes encontrado um erro, aumenta-se a possibilidade de se ter um custo menor para a solução. Um outro problema encontrado na proposta de Coelho (2007) está na falta de um planejamento detalhado para a atividade de teste, exigido para que se alcance uma boa qualidade no processo de teste.

Essa falta de planejamento não exigiu dos cooperados um conhecimento específico sobre teste e, após ter sido realizado um levantamento dentro da cooperativa, foi constatado que a atividade de teste não vem sendo realizada da forma que foi proposta.

Foi realizada uma pesquisa com o objetivo de melhorar a atividade de teste do processo de desenvolvimento de software da cooperativa. Foram encontrados na literatura alguns conceitos e metodologias sobre teste, mas poucas aplicações práticas para certificar sua eficácia.

Rios & Moreira (2006) citam um modelo de processo de teste que distribui as atividades de teste em etapas bem definidas, pois ele estabelece, para cada uma, a finalidade, os artefatos de entrada e saída e as atividades que devem ser realizadas. A Figura 4.2 mostra o ciclo de vida desse modelo.

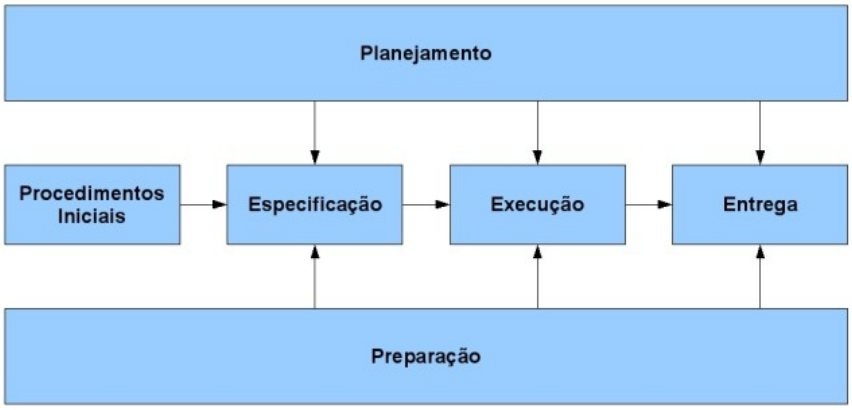

Figura 4.2: Ciclo de Vida do Processo de Teste (FONTE: Rios & Moreira, 2006)

Quatro das etapas estão em seqüência ou em cascata (Procedimentos Iniciais, Especificação, Execução e Entrega) e duas em paralelas (Planejamento e Preparação).

A primeira etapa do processo de Rios & Moreira (2006) é **Procedimentos Iniciais**. Nela, é realizado um estudo dos requisitos do negócio e é elaborado um plano com as principais atividades de teste a serem executadas, com as necessidades de recursos e ambiente, além de cronograma do processo.

A etapa **Planejamento** consiste em elaborar uma Estratégia de Teste e o Plano de Teste para servirem como fornecedor dos caminhos para as próximas etapas. Essa etapa permanecerá ativa até que o projeto seja concluído, pois uma avaliação constante sobre os rumos do projeto é necessária.

Assim como a etapa anterior, a etapa **Preparação** ocorrerá de forma paralela às outras etapas do ciclo, pois o objetivo básico desta etapa é preparar o ambiente de teste, isto é, deixar sempre disponível a infraestrutura e as ferramentas para as atividades de teste.

Na etapa **Especificação**, são elaborados os Casos de Teste e os Procedimentos de Teste.

Na etapa **Execução**, são realizados os testes propriamente ditos. Os testes são executados de acordo com os Casos de Teste e os Procedimentos de Teste. Cada resultado obtido deve ser registrado e os erros encontrados deverão ser solucionados.

A última etapa, **Entrega**, é a conclusão do projeto de testes, na qual são relatadas as ocorrências do projeto consideradas relevantes para a melhoria do processo.

Apesar de ser um modelo bem definido, o processo de teste proposto por Rios & Moreira (2006) é complexo e não atende as necessidades da cooperativa, pois tem um ciclo muito extenso, contrapondo a idéia de um processo de desenvolvimento de software mais ágil, como é o implantado na cooperativa. Além disso, não cita quem deve estar envolvido em cada uma das etapas, e não cita, durante a execução, quando as técnicas de teste devem ser utilizadas.

Um outro processo de teste encontrado durante a pesquisa foi utilizado para embasar a proposta que foi elaborada no presente trabalho. Silva (2006) definiu o SwTest, um processo de teste para uma pequena empresa. A idéia para definição do processo surgiu a partir dos problemas que a empresa na qual ele seria implantado tinha com a atividade de teste. Tais problemas são semelhantes aos encontrados na TecnoLivre, como atividade de teste mal organizada, sendo realizada apenas no fim do processo de desenvolvimento de software, execução de testes de forma aleatória e grande quantidade de retrabalho.

Silva (2006) propôs um processo mais simples, realizado em etapas, que são: Planejamento, Projeto, Execução e Avaliação. Para cada uma das etapas, foram definidos os objetivos, os artefatos, as atividades e os papéis envolvidos. Essa definição de cada uma das etapas do SwTest é semelhante às etapas do modelo de Rios & Moreira (2006) e foi além ao definir os papéis que participam de cada uma delas.

No trabalho realizado por Silva (2006), no entanto, não é apresentado o processo de desenvolvimento de software da empresa. A definição de um processo de teste leva em consideração o processo de desenvolvimento, por isso, ele deveria constar no trabalho para saber como é a relação entre os dois processos. Além disso, o processo proposto por Silva (2006) tem uma importante dependência da ferramenta Mantis (sistema livre para gerenciamento de *bugs*). Apesar de ser um facilitador para o controle dos erros ocorridos durante o projeto, a definição das ferramentas para automação do processo deveria ser realizada após o processo ser implantado na empresa.

4.2 Definição do Projeto

Com base nos trabalhos citados anteriormente, propõe-se aqui um processo de teste da cooperativa com ênfase na mudança na estrutura organizacional da cooperativa e no detalhamento das fases do processo de teste. Ao final, é mostrado novamente o fluxograma do processo de desenvolvimento de software da cooperativa, adaptado com as atividades de teste.

4.2.1 Os Papéis

A primeira etapa para a concepção do processo consiste definir os novos papéis que irão surgir com o processo de teste. A cooperativa não possui uma estrutura organizacional hierárquica. Os cooperados podem trabalhar no desenvolvimento de qualquer projeto, desde que tenham capacitação para desenvolver o projeto de acordo com suas características, como linguagem utilizada, arquitetura, etc.

A alocação da equipe para cada projeto é decidida em conjunto pelos cooperados. A partir do momento que a equipe é definida, os integrantes definem o papel de cada um. Por causa da estrutura da organização, não há profissionais específicos para cada área de TI, todos podem desempenhar o papel de gerente de projeto, analista de sistema, projetista de banco de dados e programador. O gerente é o único que deve ser realmente definido, pois é o representante do projeto perante aos outros membros da cooperativa e ao cliente. Porém, as atividades de análise de requisitos, implementação, teste e implantação são realizadas por todos.

O processo de teste requer que exista pelo menos um gerente de projeto de teste, um arquiteto de teste, um analista de teste e um testador. Foi definido que o gerente de cada projeto também ficará encarregado de ser o gerente de projeto de teste. Os outros papéis relacionados a teste ficarão a cargo de uma mesma pessoa, que deverá ser independente do projeto, isto é, não participará de qualquer outra atividade, como análise de requisitos e implementação. Isso é exigido, pois uma pessoa, com uma visão externa ao projeto, desempenhará de forma mais eficiente esse papel. No entanto, o programador pode ser um testador, realizando teste de unidade e teste de integração.

A nova estrutura da equipe de projeto, incorporada dos novos papéis, pode ser vista na Figura 4.3.

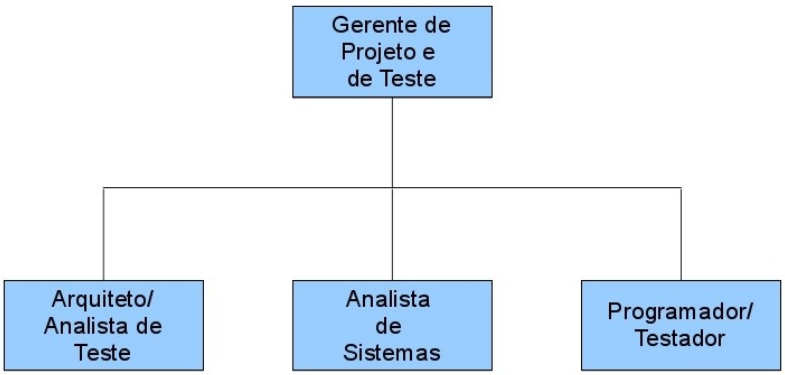

Figura 4.3: Nova Estrutura Organizacional Para os Projetos

Desta forma estrutural, os novos papéis serão responsáveis por:

- **Gerente de Projeto e de Teste:** negociar com o cliente, planejar, controlar o projeto, planejar e controlar o projeto de teste;
- **Arquiteto/Analista de Teste:** montar a infra-estrutura de teste, modelar e elaborar os casos de teste, elaborar procedimentos e executar os testes finais;
- **Analista de Sistemas:** analisar os requisitos do projeto, modelar os casos de uso, modelar banco de dados e definir arquitetura;
- **Programador/Testador:** implementar o modelo de projeto proposto e executar os testes iniciais.

Para implantar o processo de teste é importante que haja um treinamento dos cooperados para que saibam como funciona o processo quando ele começar a ser utilizado nos projetos da cooperativa.

4.2.2 Atividades do Processo

O processo de teste da TecnoLivre foi dividido em cinco atividades: (i) Planejamento do Teste, (ii) Projeto do Teste, (iii) Execução(1) do Teste, (iv) Execução(2) do Teste, (v) Finalização do Teste. As atividades são executadas ao longo do processo de desenvolvimento de software, existente na organização. A escolha do uso do termo atividade se deve a melhor adequação do processo de teste no ciclo de vida do processo, mostrado na Figura 4.1.

Este processo de desenvolvimento foi definido em cinco fases: Requisitos, Análise, Implementação, Testes, Implantação. Algumas características importantes foram levadas em consideração para a definição do processo de teste. O processo é simples e objetivo, focado na documentação e na definição do papel de cada participante no projeto. Foi proposto que o processo de teste também tivesse essas características. Assim, as atividades de teste recebem e

geram artefatos e envolvem papéis. Estes artefatos podem ser documentos, diagramas, banco de dados, ou código do programa. Os papéis envolvidos foram descritos anteriormente, na Subseção 4.2.1.

As atividades de teste, bem como os documentos gerados, foram baseados na NBR ISO/IEC 12119 e na Norma IEEE 829, descritas resumidamente no Capítulo 3.

A Figura 4.4 apresenta o fluxo das atividades no qual o processo foi dividido.

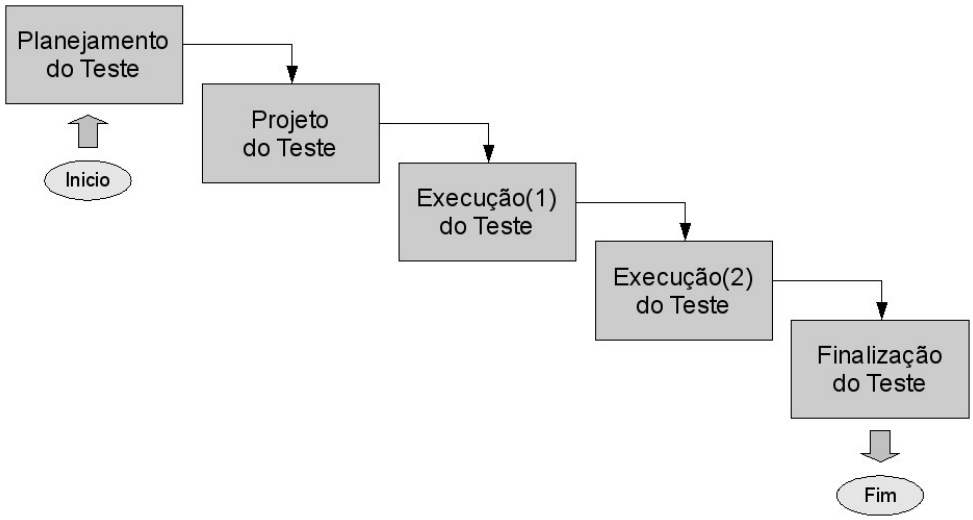

Figura 4.4: Atividades do Processo de Teste

As atividades de teste são desenvolvidas paralelamente com as atividades de desenvolvimento. Cada uma das atividades de teste foi encaixada nas fases de desenvolvimento, citadas, conforme mostra a Tabela 4.1.

Tabela 4.1: Fase do Processo de Desenvolvimento e Atividade de Teste Correspondente

Fase do Processo de Desenvolvimento	**Atividade de Teste**
Requisito	Planejamento do Teste
Análise	Projeto do Teste
Implementação	Execução(1) do Teste
Teste	Execução(2) do Teste
Implantação	Finalização do Teste

A atividade de execução do teste foi separada em duas atividades consecutivas, pois alguns testes são executados enquanto um software é implementado, como o teste de unidade e o teste de integração. Estes testes fazem parte da atividade Execução(1) do Teste. Outros testes, como o teste de validação e de sistema, estão dentro da atividade de Execução(2) do

Teste, pois devem ser executados após terem sido integrados os módulos e os componentes do programa.

A seguir, são detalhadas as atividades pertencentes ao processo de teste. Ao final de cada uma, haverá um resumo, onde define-se:

(a) os objetivos: qual a finalidade da atividade;

(b) os artefatos: os artefatos de entrada e saída da atividade;

(c) os papéis envolvidos: as pessoas que participam da atividades;

(d) os passos executados: os procedimentos que serão realizados.

Após a definição, será mostrado na Figura 4.5 o processo de desenvolvimento de software da cooperativa com o processo de teste integrado.

(i) Planejamento do Teste

Na atividade Planejamento do Teste, deve ser definido o escopo do teste, isto é, define-se o que será testado durante o projeto, a metodologia de teste que será aplicada, o cronograma, o pessoal que será alocado o processo de teste e os itens que serão testados.

A atividade será realizada paralelamente com o levantamento de requisitos, na fase de Requisitos do processo de desenvolvimento. O responsável será o gerente de projeto, que ficará encarregado de levantar os requisitos e definir o plano de teste, podendo haver a colaboração do arquiteto de teste.

Como conseqüência da atividade, tem-se a geração do Plano de Teste, onde estará documentado o resultado das definições feitas na atividade.

Resumo da atividade Planejamento do Teste:

1. **Objetivos:** verificar o levantamento dos requisitos do software; definir a metodologia para o processo de teste do software a ser desenvolvido; estabelecer equipe;

2. **Artefatos:** será utilizado o documento de requisitos e a atividade resultará no Plano de Teste;

3. **Papéis Envolvidos:** Gerente de Projeto e de Teste e pode haver a colaboração do Arquiteto/Analista de Teste, dependendo do projeto a ser desenvolvido;

4. **Passos executados:**

   - Verificar as características do projeto;
   - Verificar requisitos levantados;
   - Definir itens a serem testados e não testados;
   - Definir a estratégia a ser aplicada;
   - Definir as técnicas a serem utilizadas;
   - Definir cronograma;
   - Alocar o pessoal para o processo de teste;
   - Definir as responsabilidades de cada pessoa envolvida;
   - Realizar treinamento do pessoal, se necessário, para utilização de ferramentas de automação da atividade de teste;
   - Definir o ambiente necessário para o teste (software e hardware);
   - Definir as ferramentas de teste;
   - Identificar os possíveis documentos gerados pelas atividades de teste;
   - Gerar Plano de Teste.

(ii) Projeto do Teste

Durante a atividade de Projeto do Teste, são detalhados como será o teste e a forma como será implementado. Serão detalhados os itens a serem testados e os procedimentos a serem seguidos para testá-los. Para a execução dessa atividade, será feita a revisão dos documentos gerados durante a fase de Análise do processo de desenvolvimento. Essa revisão pode ser considerada um teste, pois poderá verificar a existência de erros nesses documentos.

A atividade Projeto do Teste será realizada paralelamente as atividades de criação do diagrama de casos de uso e do diagrama de classes, definição da arquitetura e modelagem do banco de dados, durante a fase de Análise. O arquiteto/analista será o encarregado da atividade, podendo haver colaboração do gerente de projeto e de teste, do analista de sistema e do testador, que pode ser o possível desenvolvedor do projeto.

O documento gerado nessa atividade será o Projeto do Teste, onde constará os detalhes da atividade, os Casos de Teste e o Procedimento de Teste para cada um deles. Em cada Caso de Teste, deve-se determinar as entradas e saídas necessárias para a sua execução e o ambiente

necessário, além de identificar a dependência para outro caso de teste, caso haja. O Procedimento de Teste descreve os passos necessários para a realização de um conjunto de Casos de Teste.

Resumo da atividade Projeto do Teste:

1. **Objetivos:** detalhar a bateria de testes; definir e detalhar os Casos de Teste; definir e detalhar os procedimentos de teste para cada conjunto de caso;

2. **Artefatos:** serão utilizados o diagrama de caso de uso, o diagrama de classes, o documento da arquitetura e a modelagem do banco de dados. A atividade gerará o documento Projeto de Teste, onde constará os Casos de Teste e o Procedimento de Teste;

3. **Papéis Envolvidos:** o Arquiteto/Analista de Teste é o principal envolvido, podendo haver colaboração do Gerente de Projeto e de Teste, Analista de Sistema e do Programador/Testador;

4. **Passos executados:**

   - Verificar o Plano de Teste;
   - Verificar e revisar os documentos gerados na fase de Análise;
   - Reportar erros nesses documentos, caso hajam;
   - Detalhar os itens a serem testados;
   - Detalhar a estratégia de teste escolhida;
   - Detalhar as técnicas de teste a serem aplicadas em cada item;
   - Definir os casos de teste;
   - Definir os procedimentos de teste;
   - Preparar o ambiente para cada caso de teste;
   - Gerar documento de Projeto de Teste, contendo os Casos de Teste e o Procedimento de Teste para cada um dos Casos.

(iii) Execução(1) do Teste

Essa atividade consiste na execução dos primeiros testes planejados nas fases anteriores. Serão utilizados os Casos de Teste e os Procedimentos de Teste gerados na atividade anterior para executar esses primeiros testes.

A atividade Execução(1) do Teste será realizada durante a fase de Implementação do processo de desenvolvimento, paralelamente às atividades de implementação do programa e do banco de dados.

Poderão ser aplicados testes de unidade, em cima dos primeiros códigos implementados, sendo aplicados pelo próprio Programador. Poderão ser verificados no software: as estruturas de dados, as condições limites, o tratamento de erros, entre outras características do projeto que requeiram ser testadas.

Os testes de integração também poderão ser aplicados, a partir do momento que módulos do sistema tenham sido implementados e liberados para a integração. Ambos os testes podem ser executados também pelo próprio Programador e/ou especialista de teste (Arquiteto/ Analista de Teste) da equipe do projeto.

Um artefato gerado nessa atividade é o Registro de Teste. Nele, constará o relatório diário dos testes, no qual é relatado cronologicamente o procedimento realizado e seu respectivo Caso de Teste. Também constará o relatório de incidentes que possam ocorrer durante o teste, para ser analisado posteriormente. As informações contidas no Registro de Teste serão importantes para a geração do Relatório Final do Teste, visto mais adiante.

Caso alguma implementação não esteja em conformidade com os requisitos levantados, cabe ao executor do teste encaminhar o problema detectado para o especialista de teste do projeto entrar em contato com o responsável pelo levantamento de requisitos para que se encontre a solução. Não havendo erros na implementação, uma versão intermediária do software é gerada para a atividade Execução(2) do Teste, detalhada a seguir.

Resumo da atividade Execução(1) do Teste:

1. **Objetivo:** executar os primeiros testes na implementação para avaliar sua conformidade com os requisitos;

2. **Artefatos:** serão utilizados o Projeto de Teste, onde constará os Casos de Teste e Procedimentos de Teste, e as implementações do software. Como resultado, será gerada uma versão intermediária do software para testes posteriores e o Registro de Teste com os primeiros relatórios do teste;

3. **Papéis Envolvidos:** o Programador é o principal envolvido, assim como Analista/Arquiteto de Teste, na execução dos testes, sendo o segundo responsável em reportar erros ao Analista de Sistemas;

4. **Passos executados:**

- Analisar os Casos de Teste e Procedimentos de Teste;
- Executar testes de unidade nas primeiras implementações;
- Executar testes de integração para verificar conformidade com os requisitos do software;
- Relatar diariamente os procedimentos realizados para a execução do teste;
- Registrar incidentes que possam ocorrer durante a execução do teste;
- Analisar resultados do teste;
- Relatar, ao analista de sistema, problemas encontrados nas implementações, caso ocorram;
- Encaminhar a versão intermediária do software para a atividade de teste seguinte.

(iv) Execução(2) do Teste

Essa atividade é a segunda etapa da execução dos testes de software no processo. Os testes realizados serão baseados nos Casos de Teste e Procedimentos de Teste gerados durante a atividade Projeto do Teste.

A atividade Execução(2) do Teste será realizada durante a fase de Teste do processo de desenvolvimento da cooperativa, após a fase de Implementação. Decidiu-se por manter essa fase, por ser ponto chave no processo de desenvolvimento atual da cooperativa, pois o software só segue para a implantação, se for aprovado nessa fase.

O software testado estará na sua versão intermediária e deve estar como um pacote dos módulos, não contendo mais erros relacionados a codificação, banco de dados e interfaces.

Serão executados os testes de validação e de sistema. O teste de validação será aplicado para validar os requisitos do software, isto é, se ele realiza as funções que foram especificadas durante o levantamento dos requisitos. O teste de sistema será realizado para verificar a interação do software com os outros elementos do sistema para o qual ele foi desenvolvido. Este teste poderá ser realizado no ambiente do sistema ou na própria cooperativa, desde que consiga simular o ambiente no qual será implantado.

Ambos os testes devem ser realizados pelo Arquiteto/Analista de Teste do projeto, podendo haver a colaboração do Programador que desempenha o papel de Testador. Outra pessoa que pode participar desses testes é o cliente.

As execuções serão relatadas no Registro de Teste gerado na atividade anterior. Portanto, devem ser realizados os mesmos procedimentos para a geração do documento.

Se durante o teste for encontrado algum erro, ou seja, não estiver atendendo aos requisitos levantados, o especialista deve reportá-lo ao Analista de Sistema, assim como é realizado na atividade Execução(1) de Teste. Se o software estiver em conformidade com os seus requisitos, ele então estará apto para ser implantado no ambiente para o qual foi projetado.

Resumo da atividade Execução(2) do Teste:

1. **Objetivo:** executar os testes para verificação e validação do software em relação aos requisitos levantados;

2. **Artefatos:** serão utilizados o Projeto de Teste, onde constará os Casos de Teste e Procedimentos de Teste, a versão intermediária do software e o Registro de Teste. Serão gerados o Registro de Teste modificado, e a versão final do software para a implantação;

3. **Papéis Envolvidos:** o Arquiteto/Analista de Teste será o principal envolvido na atividade, mas poderá contar com a colaboração do Programador e do cliente;

4. **Passos executados:**

   - Analisar os Casos de Teste e Procedimentos de Teste;
   - Executar testes de validação para verificar as conformidades do produto com os seus requisitos;
   - Executar testes de sistemas para verificar a compatibilidade do software com o sistema para o qual foi desenvolvido;
   - Relatar diariamente os procedimentos realizados para a execução do teste;
   - Registrar incidentes que possam ocorrer durante a execução do teste;
   - Analisar resultados do teste;
   - Relatar, ao Analista de Sistemas, problemas encontrados nas implementações, caso ocorram;

- Encaminhar a versão final do software para ser implantado.

(v) Finalização do Teste

A atividade Finalização do Teste consiste na realização dos últimos testes, quando o software for implantado no ambiente para o qual foi desenvolvido. Essa atividade serve para mostrar ao cliente que o software está funcionando. Alem disso, ela serve como forma de avaliação do processo de teste, pois mostrará se o processo eliminou os erros antes da entrega final ao cliente.

Essa atividade é realizada durante a fase de Implantação do processo de desenvolvimento, sendo executado enquanto o software está sendo implantado. Poderão ser executados testes de sistema, como testes de segurança, desempenho, usabilidade.

A equipe de implantação do projeto ficará a cargo da execução dos testes, podendo haver supervisão do Arquiteto/Analista de Teste. Ocorrendo algum erro durante a execução do teste, novamente, caberá ao especialista de teste da equipe de projeto reportar o problema para o Analista de Sistemas e tentar encontrar a solução, analisando novamente os requisitos do sistema. Não havendo erro algum, o software é entregue ao cliente.

Por fim, será gerado o Relatório Final de Teste, onde constarão os registros das atividades anteriores, além da avaliação final feita pelos envolvidos no processo. O objetivo é avaliar se o processo de teste foi realizado de forma correta, se atende à cooperativa e será utilizado para melhoria do processo.

Resumo da atividade Finalização do Teste:

1. **Objetivo:** certificar que o software não contém mais erros e está em conformidade com os requisitos. Avaliar o processo de teste utilizado;
2. **Artefatos:** Receberá a versão final do software e o Registro de Teste. O documento gerado na atividade será o Relatório Final de Teste;
3. **Papéis Envolvidos:** A equipe de implantação do processo de desenvolvimento, além do Arquiteto/Analista de Teste;
4. **Passos executados:**
   - Analisar os Casos de Teste e Procedimentos de Teste;
   - Executar testes de sistemas para verificar a compatibilidade do software com o sistema para o qual foi desenvolvido;

- Registrar incidentes que possam ocorrer durante a execução do teste;
- Relatar, ao Analista de Sistema, problemas encontrados nas implementações, caso ocorram;
- Analisar resultados do processo de teste e apresentá-los a toda equipe envolvida no projeto;
- Gerar Relatório Final de Teste.

4.3 Fluxograma do Processo de Desenvolvimento

A Figura 4.5 mostra o modelo de processo de desenvolvimento de software proposto por Coelho (2007), modificado pela inserção do processo de teste.

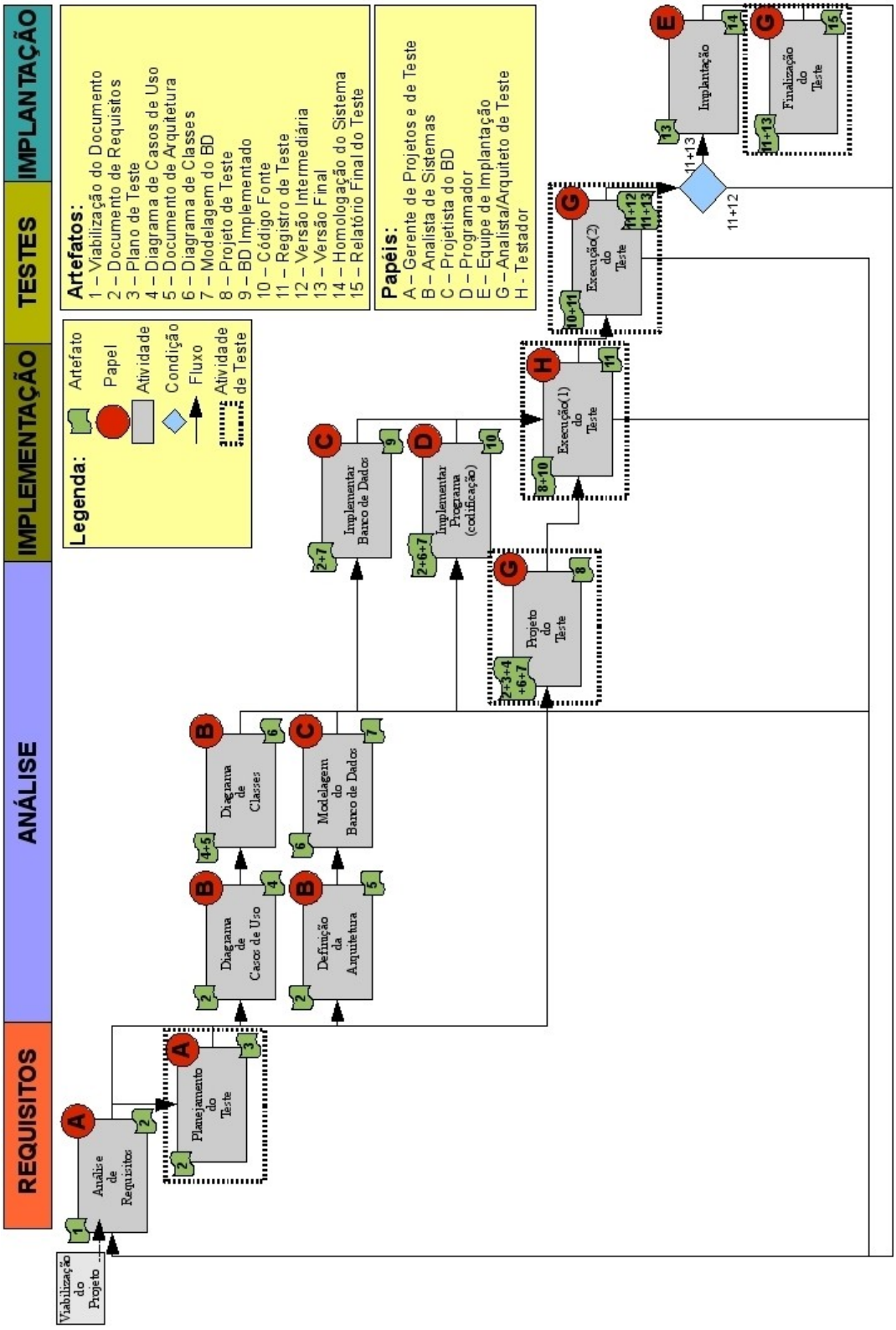

Figura 4.5: Processo de Desenvolvimento de Software (Fonte: Coelho 2007, adaptado)

Nesse modelo, foram inseridos as atividades de teste, que estão em destaque nos quadros sublinhados (Planejamento do teste, Projeto do Teste, Execução(1) do Teste, Execução(2) do Teste e Finalização do Teste), os artefatos (Plano de Teste, Projeto de Teste, Registro de Teste e Relatório Final de Teste) e os papéis (Gerente de Projeto de Teste, Analista/ Arquiteto de Teste e Testador) e foram removidos: o papel Equipe de Teste, os artefatos Relatórios de Testes (equipe) e Relatórios de Testes (cliente).

4.4 Considerações Finais

Foi elaborado um processo de teste de software que atendesse uma cooperativa de desenvolvimento que não tem implantada uma atividade de teste organizada. Além dos problemas citados, o processo de teste será utilizado para garantir que software desenvolvido seja de qualidade.

Com base em modelos existentes de processo de teste e no processo de desenvolvimento de software da cooperativa, um processo foi definido. O foco maior do processo de teste foi na mudança da estrutura, com a inclusão de novos papéis que a atividade de teste requer, e nas atividades (Planejamento do Teste, Projeto do Teste, Execução(1) do Teste, Execução(2) do Teste e Finalização do Teste) executadas durante o processo de teste, detalhando para cada uma a finalidade, artefatos, papéis envolvidos e passos a serem executados. Dessa forma, o processo não gerará impacto quando for implantado na organização para o qual foi elaborado.

5 CONCLUSÃO

Após fazer um estudo sobre a atividade de teste em geral, especificamente processo de teste e qualidade na produção de software, e fazer um levantamento sobre os problemas da atividade de teste em uma cooperativa que desenvolve software livre, definiu-se um processo de teste que se encaixa no processo de desenvolvimento de software da organização, respeitando as características deste processo, que é simples, objetivo, baseado em documentação e definição clara dos papéis envolvidos. Espera-se que, com isso, após a implantação na cooperativa, o impacto negativo seja o menor possível.

O proposta de processo de teste visou a mudança na estrutura organizacional da cooperativa, com a criação de novos papéis que participam do teste. O processo de teste definido é constituído por cinco atividades e é realizado paralelamente ao processo de desenvolvimento utilizado na cooperativa.

A utilização pelas empresas de um processo de teste bem planejado, sendo realizado desde o início do processo de desenvolvimento de software, é uma forma de garantir que o software tenha a maior qualidade possível. É mister lembrar, porém, que essa prática começou a ser aplicada de forma recente, e tem poucos resultados documentados sobre essa forma de realizar o teste.

O processo de teste definido neste trabalho, por enquanto, é uma proposta, que necessita ser implantada e avaliada. A partir disso, a cooperativa poderá dar seqüência em seu objetivo de melhorar a atividade de teste e a produção de software de qualidade.

5.1 Trabalhos Futuros

A partir deste trabalho, alguns outros podem ser desenvolvidos para chegar no objetivo final que a cooperativa deseja. Alguns deste trabalhos são:

- Utilizar o processo em um projeto piloto para avaliá-lo;
- Propor melhoria para esse processo de teste, a partir da sua avaliação;
- Fazer um estudo sobre as ferramentas de gerenciamento e de automação utilizadas em um processo de teste e, com isso, escolher as melhores para serem aplicadas no processo definido neste trabalho.

6 REFERÊNCIAS BIBLIOGRÁFICAS

BARBOSA, E. F.; MALDONADO, J. E.; VICENZI, A. M. R.; DELAMARO, M. E.; SOUZA, S. do R. S. de; JINO, M.. **Introdução ao Teste de Software** Universidade de São Paulo – ICMC/USP; Universidade Estadual de Maringá – DIN/UEM; Universidade Estadual de Ponta Grossa – UEPG; Universidade Estadual de Campinas — DCA/FEEC/UNICAMP.

BARTIE, A.. **Processo de Teste de Software**. Disponível em: <http://imasters.uol.com.br/artigo/6102/des_de_software/processo_de_teste_de_software_-_parte_01/>, acessado em 20/10/2008.

COELHO, A. C.. **Implantação de um Processo de Desenvolvimento de Software em uma Cooperativa de Software Livre** / Alexandre Chaves Coelho. Lavras – Minas Gerais, 2007. 63p.

CRESPO, A. N.; SILVA, O. J. da; BORGES, C. A.; SALVIANO, C. F.; JUNIOR, M. de T. e A. JINO, M.. Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo . In: **III Simpósio Brasileiro de Qualidade de Software(SBQS 2004).** Brasília.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software** / organização Márcio Eduardo Delamaro, José Carlos Maldonado, Mario Jino. - Rio de Janeiro: Elsevier, 2007.

GOMES, N. Da S.. **Qualidade de Software – Uma Necessidade**. Disponível em: <http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf>, acessado em 25/07/2008.

IEEE 829. **IEEE Standard for Software Test Documentation** . Institute of Electrical and Electronics Engineer, 1998.

ISO/IEC 15504-5. **Information Technology - Process Assessment Part 5: An exemplar Process Assessment Model**. International Organization for Standardization. 2006.

ISO/IEC 9126. **Information Technology – Software Product Evaluation – Quality characteristics and guidelines for their use**. International Organization for Standardization. 1991.

JUNG, C. F.. **Metodologia para Pesquisa & Desenvolvimento – Aplicada a Novas Tecnologias, Produtos e Processos**. 1.ed. Rio de Janeiro: Axcel Books do Brasil Ltda, 2004. 312 p.

MYERS, G. J.. **The art of software testing** / Glenford J. Myers ; Revisado e atualizado por Tom Badgett and Todd Thomas, com Corey Sandler. - 2.ed. - New Jersey. Wiley & Sons, Inc., Hoboken, New Jersey. 2004.

NBR ISO/IEC 12119. **Tecnologia de Informação – Pacotes de software – Teste e requisitos de qualidade**. Associação Brasileira de Normas Técnicas. 30 de outubro de 1998.

PÁDUA, W. De, P. F.. **Engenharia de Software, Fundamentos, Métodos e Padrões.** - 2.ed. - Rio de Janeiro. LTC. 2003.

PERES, G. B.. **Implantação de um Processo de Teste de Software em uma empresa de TI: Um Estudo de Caso.**

PRESSMAN, R. S.. **Engenharia de Software.** / Roger S. Pressman; tradução Rosângela Delloso Penteado, revisão técnica Fernão Stella R. Germano, José Carlos Maldonado, Paulo Cesar Mariero -6.ed. - São Paulo : McGraw.Hill. 2006.

RIOS, E.; MOREIRA, T.. **Teste de Software 2ª edição revisada e ampliada.** / Emerson Rios; Trayahú Moreira Filho. - 2.ed Rio de Janeiro. Alta Books. 2006. 224p.

ROCHA, A. R.. **Modelo de Referência para melhoria de processo do software (MR-MPS).** Disponível em: <http://golden.softex.br/portal/softexweb/uploadDocuments/MPSBR-Modelo-de-Referencia.pdf>, acessado em 04/10/2008.

SALVIANO, C. F.. **Melhoria e Avaliação de Processo de Software com o Modelo ISO/IEC 15504-5:2006** / Clênio Figueiredo Salviano. – Lavras : UFLA, 2006. 213 p. : il. – Curso de Pós-Graduação "Lato-Sensu" (Especialização) a Distância: Melhoria de Processo de Software.

SALVIANO, C. F.. **Uma proposta orientada a perfis de capacidade de processo para evolução da melhoria de processo de software** / Clênio Figueiredo Salviano. - Campinas, SP : [s.n.], 2006.

SILVA, V. M. de P.. **SwTest: Um Processo de Teste Definido para uma Empresa de Pequeno Porte Desenvolvedora de Software** / Vanessa Maira de Paiva Silva – Minas Gerais, 2006. 55p.

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

VILLAS BOAS, A. L. C.. **Qualidade e Avaliação de Produto de Software** / André Luiz de Castro Villas Boas. - Lavras : UFLA/FAEPE, 2007. 141 p. : il. – Curso de Pós-graduação "Lato Sensu" (Especialização) a Distância - Melhoria de Processo de Software.

WANGENHEIN, C. G. Von; PICKLER, K. K.; THIRY, M.; ZOUCAS, A.C.; SALVIANO, C. F.. Aplicando Avaliações de Contextualização em Processos de Software Alinhados ao CMMI-SE/SW . In: **Simpósio Internacional de Melhoria de Processo de Software (SIMPROS 2005).** São Paulo.