FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” – UNIVEM
CURSO DE ADMINISTRAÇÃO

ANDERSON TADAYOSHI FUKUMORI
ISMAÍAS GRACINO DOS SANTOS
TIAGO MAY MORRO

A IMPORTÂNCIA DA ATIVIDADE DE TESTE NO DESENVOLVIMENTO DE SOFTWARE

MARÍLIA
2008

ANDERSON TADAYOSHI FUKUMORI
ISMAÍAS GRACINO DOS SANTOS
TIAGO MAY MORRO

A IMPORTÂNCIA DA ATIVIDADE DE TESTE NO DESENVOLVIMENTO DE SOFTWARE

Trabalho de curso apresentado ao Curso de Administração da "Fundação de Ensino Eurípides Soares da Rocha" mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito para obtenção do grau de bacharel em Administração com linha de formação em Análise de Sistemas.

Orientador:
Prof. Ms. Adalberto Sanches Munaro

MARÍLIA
2008

FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

Mantenedora do Centro Universitário Eurípides de Marília - UNIVEM

Cursos: Administração de Empresas, Análise de Sistemas, Comércio Exterior, Marketing.

Anderson Tadayoshi Fukumori - 35067-2
Ismaías Gracino dos Santos - 35502-1
Tiago May Morro - 34212-2

TÍTULO "A IMPORTÂNCIA DA ATIVIDADE DE TESTE NO DESENVOLVIMENTO DE SOFTWARE "

Banca examinadora do Trabalho de Curso apresentada ao Programa de Graduação em Administração de Empresas da UNIVEM, F.E.E.S.R, para obtenção do Título de Bacharel em Administração de Empresas.

Nota: 9,0

ORIENTADOR: _____
Adalberto Sanches Munaro

1° EXAMINADOR: _____
Ricardo Petruzza do Prado

2° EXAMINADOR: _____
Jose Eduardo Santarem Segundo

Marília, 29 de novembro de 2008.

Primeiramente a Deus, que nos presenteou com a vida e a oportunidade de realizamos esse trabalho;

A nossos pais, já que são aqueles que ajudam a nos colocar de pé quando nossas asas esquecem como voar;

Aos amigos adquiridos ao longo do curso, pelas contribuições e o apoio.

AGRADECIMENTOS

Agradecemos aos docentes e colegas da instituição, que contribuíram para a elaboração deste trabalho desde o surgimento da idéia.

À instituição de ensino UNIVEM, que dispôs do suporte necessário para as pesquisas que fossem realizadas para inserção no trabalho.

Em especial, agradecemos ao professor Adalberto Sanches Munaro, que nos orientou, contribuindo no desenvolvimento e conclusão deste trabalho.

"Bem aventurado o homem que acha sabedoria,

e o homem que adquire conhecimento"

Salomão

FUKUMORI, Anderson Tadayoshi; SANTOS, Ismaías Gracino; MORRO, Tiago May. **A Importância da Atividade de Teste no Desenvolvimento de Software**. 2008. 66 f. Trabalho de Curso de Administração – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2008.

RESUMO

A presente monografia aborda assuntos relativos à importância dos métodos e processos, atualmente utilizados para atividade de teste no desenvolvimento de software. Apresentando primeiramente a fase de teste dentro da estrutura de desenvolvimento de software, demonstrando seu conceito e representando sua importância dentre as outras fases. Independente do modelo de desenvolvimento a evolução da fase de teste, com a criação de novas técnicas tem sido de grande importância para obter maior qualidade no produto. Com um mercado ainda mais exigente pela qualidade, não seria diferente com relação aos produtos de tecnologia. No desenvolvimento de software, dar importância a fase de atividade de teste, é priorizar a qualidade e confiabilidade, resultando em melhorias da imagem da organização desenvolvedora.

Palavras-Chave: Requisitos. Desenvolvimento. Teste. Software. Qualidade. Confiabilidade.

FUKUMORI, Anderson Tadayoshi; SANTOS, Ismaías Gracino; MORRO, Tiago May. **A Importância da Atividade de Teste no Desenvolvimento de Software**. 2008. 66 f. Trabalho de Curso de Administração – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2008.

ABSTRACT

The present monograph talking relatives subjects to the importance of methods and process actually, used to actvit of test, and in development of software. First of all, showing the test fase inside the strut we of development of the software, showing the concept and representing the importance among the other fase. Independent of the development model the fase's evolution will the action of new tecnitione has been a big importance to obtem a bigger quality in the product. With a market the more and more demanding by quality wouldn't be different with the relation to the technologic product. In software's development, giving importance to the act wit's faze of test, is priories the quality and confinable resulting in a image's improvement in the developer organization.

Keywords: Requirements. Development. Test. Software. Quality. Confinable.

FUKUMORI, Anderson Tadayoshi; SANTOS, Ismaías Gracino; MORRO, Tiago May. **A Importância da Atividade de Teste no Desenvolvimento de Software**. 2008. 66 f. Trabalho de Curso de Administração – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2008.

RESUMEN

La presente monografía aborda temas relativos a la importancia de los métodos y procesos usados actualmente para en la actividad para testar el desarrollo del software. Primeramente se presentará el test hecho dentro de su estructura del desarrollo del software, demostrando su concepto y representar su importancia en las otras fases. Independiente del modelo de construcción, la evolución de la fase de test, con la creación de nuevas técnicas ha sido de gran importancia para obtener mayor calidad en el producto. Con el mercado cada día más exigente por calidad, dar importancia a la fase de actividad del test, es dar prioridad a la calidad y a la confiabilidad, con resultado en mejorías del imagen de la organización desarrolladora.

Palabras Claves: Requisitos. Desarrollo. Test. Software. Calidad. Confiabilidad.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ALATS: Associação Latino-Americana de Teste de Software

CBTS: Certificação Brasileira de Teste de Software

CMST: Certified Manager of Software Testing

CQT: Controle de Qualidade Total

CSTE: Certified Software Tester

CSTP: Certified Software Test Professional

CTM – Certified Test Manager

IEC: International Eletrotechnical Commission

IIST: International Institute for Software Testing

ISO: International Standardization Organization

ISTQB: International Software Testing Qualifications Board

ITG: Independent Test Group

NBR: Normas Brasileiras

QAI: Quality Assurance Institute

TI: Tecnologia da Informação

LISTA DE GRÁFICOS

SUMÁRIO

INTRODUÇÃO

A atividade de Teste de Software é uma fase do ciclo de vida de um sistema que consiste em testar o produto (software) através de métodos estabelecidos, a fim de eliminar erros que possam ter surgido de outras fases do desenvolvimento, erros esses que se não percebidos, causam muitos malefícios à organização que conta com o software para seus processos, e ao próprio desenvolvedor.

Este trabalho tem como objetivo principal enfatizar a importância dos métodos, dos processos, das técnicas, dos profissionais, entre todos os outros pontos que envolvem Atividade de Teste de Software. O motivo desta demonstração é a grande quantidade de livros e trabalhos que apresentam todo o conceito de teste, ou seja, livros de conteúdo técnico que não frisam o tema "importância de teste" com o intuito de tornar essa atividade mais comum no mercado, e com o objetivo de apresentar os benefícios que são agregados ao software quando essas técnicas são aplicadas corretamente. O trabalho apresenta também, o mercado atual na área de Teste de Software.

Assim como a utilização de sistemas e a criação de software tem crescido significativamente nos últimos anos, a qualidade exigida vem nesse ritmo acelerado. Em se tratando de bens e serviços que estão sendo controlados por meio de sistemas, ou seja, todas as informações de extrema importância para tomada de decisão, assim como a locação da responsabilidade das organizações são dependentes de um sistema, a qualidade vem crescendo, assim as normas e exigências do mercado estão cada vez maiores. Por isso, os processos e as normas para se ter um software confiável, torna indispensável no processo de desenvolvimento de software, os métodos, técnicas, e ferramentas que permitem a realização da atividade de teste de maneira sistematizada e com fundamentação científica, de modo a aumentar a produtividade e a qualidade.

Além da qualidade de software, a atividade de teste traz outros benefícios que não envolvem somente o lado técnico do assunto, mas também questões gerenciais, como custo financeiro do projeto, agiliza a construção do mesmo, ocorre a diminuição ou exclusão de manutenção do sistema, satisfação do usuário (ou cliente) e conseqüentemente reconhecimento da qualidade que a organização desenvolvedora pode apresentar em outros projetos.

CAPÍTULO 1 – CICLO DE VIDA DO SOFTWARE

Para melhor abordagem da importância da fase de testes no desenvolvimento de um software, deve ser claro o entendimento do ciclo de vida de um software, ou seja, o seu desenvolvimento. É nesse ciclo de vida que está a fase chamada Teste onde é realizada antes de ser implantada. Esta fase possui a sua importância, o seu custo, seus diversos métodos, e as mais variadas opiniões.

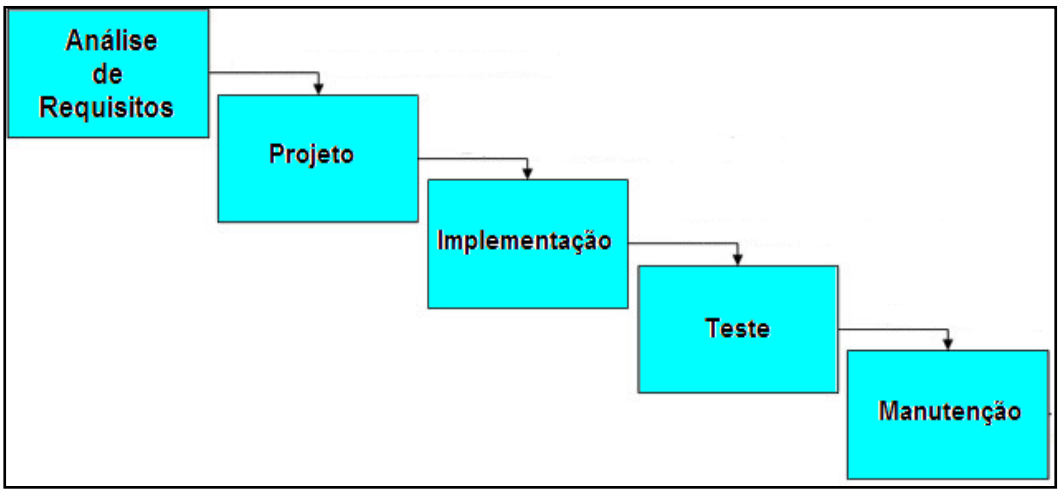
Um software para ser construído e utilizado ele é fundamentado em fases desde o início de seu desenvolvimento até sua utilização final, essas fases têm como objetivo organizar o seu desenvolvimento para que o sistema possa atingir o objetivo esperado, determinado pelo cenário em que ele irá atuar.

Para conceituar o ciclo de vida de um sistema foi utilizado nesse trabalho modelo de desenvolvimento de software, chamado “Cascata”, existem outros modelos que podem ser citados, porém menos comuns no ciclo de vida de um sistema.

O modelo Cascata é um modelo linear em que cada passo deve ser completado antes que o próximo possa ser iniciado. O modelo inicia com a primeira fase Análise de Requisitos e percorre as outras fases, Projeto, Implementação, Teste e Manutenção, cada etapa com as suas características de desenvolvimento que no final originam um software. Este modelo pode ser interpretado de várias formas com relação a cada fase, por isso, podem variar em alguns aspectos, mas fundamentalmente não é drasticamente alterado.

A figura 1 mostra a especificação de cada fase do ciclo de vida de um sistema, em especial a fase de Teste.

Figura 1- Modelo de desenvolvimento Cascata



1.1 Análise de Requisitos

Requisitos são descrições de funções e/ou serviços que serão oferecidos pelo sitema, com base em informações que refletem as necessidades do cliente. A analise de Requisitos não fica somente na necessidade do cliente que procura resolver algum problema, controlar um dispositivo ou encontrar informações. A analise de requisitos descreve todas possibilidades de fluxo de informações e busca encontrar lacunas no sistema atual, tornando mais viável o sistema a ser desenvolvido. Assim na solicitação de um cliente para a construção de um novo sistema, o cliente tem a noção do que o sistema fará. O novo sistema, tanto para substituição de um existente ou apenas um aprimoramento/extensão de um sistema atual (manual ou automático), é analisado no novo sistema proposto, um planejamento para tarefas que nunca haviam sido realizadas antes.

Pfleeger (2004, p. 111) define requisitos como “descrição de algo que o sistema é capaz de realizar para atingir os seus objetivos”.

> Uma compreensão completa dos requisitos de software é fundamental para um bem-sucedido desenvolvimento de software. Não importa quão bem projetado ou quão bem codificado seja, um programa mal analisado e especificado desapontará o usuário e trará aborrecimentos ao desenvolvedor (PRESSMAN, 1995, p. 231).

Análise de Requisitos é um processo de descobrir e criar as funções, documentações e restrições de um sistema. Esse processo é chamado de “Requirements Engineering” (Engenharia de Requisitos).

A análise de requisitos bem como sua especificação, o conhecimento detalhado sobre os tipos de requisitos, deteminam as caracteristicas do sistema, assim o desenvolvedor tem a noção exata para avaliar a qualidade, viabilidade, tempo e as dificuldades do software logo no inicio da construção.

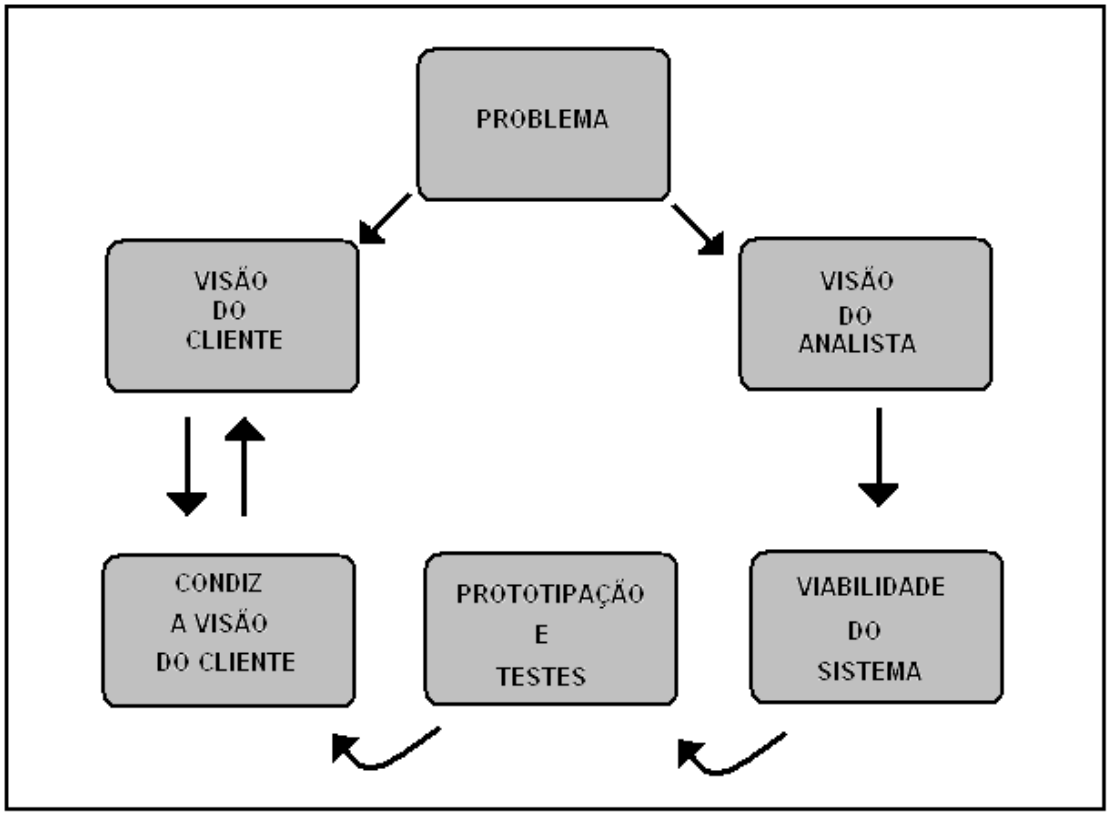
1.1.1 Identificação de Requisitos

Conhecer os requisitos sem duvida é uma tarefa importante para atingir as necessidades impostas pelo usuario/cliente. Porém como vimos anteriormente, a visão imposta pelo cliente retrara uma necessidade de resolver um problema como vemos na

Figura 2 . Porém essa visão é vaga e a identificação dos tipos de requisitos vai além de um problema. Deve-se identificar não somente o problema, mas também as pessoas, os processos e recursos envolvidos e assim documentar as relações entre eles. Fazendo a mesma pergunta de formas diferente para ter a certeza de que foi entendido o que os usuários/clientes precisam.

Determinado os requisitos, conseguimos descrever o comportamento de um sistema. Sendo os Requisitos Funcionais, que faz a relação e a interação entre o sistema e seu ambiente, descrevendo o que o sistema deve fazer e Requisitos não Funcionais, que descrevem as restrições no sistema são aqueles não diretamente relacionados as funções específicas.

Figura 2 – Identificação do Problema



A documentação definindo a especificação de requisitos descrevem as formas que o sistema interagirá com o ambiente. Essa documentação inclui os seguintes tipos de itens:

- **Ambiente Físico:** colocação dos equipamentos e a rede que ligará esses equipamentos em vários lugares e setores da organização. É importante o analista descrever as condições que se trabalhará com o sistema, em caso de não conformidade informar que o sistema não funcionará por determinado motivo, que pode ser temperatura, interferencia, umidade e outros;

- **Interface:** importante descrever todas as interfaces que estão no sistema atual, necessário saber se as entradas de informações tem origens em não conformidade com o sistema que será implantado, ou seja, entrada de informação recebida de uma balança industrial, como o sistema entenderá esse tipo de informação? Quais as linguagens utilizadas no sistema atual que permanecerá? Os sistemas operacionais? Série de dúvidas que obrigatoriamente devem ser tiradas na análise de requisitos;

- **Os usuários:** importante planejar a facilidade que o sistema deverá obter visto as habilidades dos funcionários, já nesta etapa se programa a maneira correta de treinamento, com base nas dificuldades que estão previstas na compreensão dos usuários com o novo sistema. Quais as restrições que obrigatoriamente deverão conter no sistema?

- **Funcionalidade:** determinar a maneira e o tempo que o sistema realizará as funções desejadas, também a maneira correta que o sistema irá operar, e as manutenções e modificações que deixarão o sistema próximo do ideal;

- **Documentação:** as ajudas poderão ser on-line, em formato de livro, apresentação, CD. Analisar com a fase de estudos dos usuários, qual será o tipo de público que destinarão os tipos de documentação;

- **Dados:** junto com a fase de interfases deve estudar qual os tipos de dados será utilizados para entrada e saída de informação, a necessidade de manter e armazenar esses dados de forma eficaz e qual precisão deve ter os relatórios e todo tipo de informação declarado pelo usuário;

- **Recursos:** estuda-se nessa fase a importancia da analise dos materiais envolvidos, se há a necessidade de investimentos em hardware, definido o tempo e habilidades para desenvolver o requerido;

- **Segurança:** trabalhando em rede, em sistemas de proporções imensas, é necessário organizar quais os tipos de informações será manipulada, e qual as permissões e restrições dada aos usuários. A forma de controlar os acessos isolando informações de um usuário e permitindo a outro. De forma que um operador de caldeira, não consiga visualizar informações do setor de contabilidade. Também determinamos como será os (Backup) e onde armazenará, ou enquantos locais armazenará;

- **Garantia da qualidade:** estuda-se os tempos das falhas, o que é aceitável e como será resolvido os problemas encontrados. Fase próximo a um Feedback do sistema, onde todas as duvidas e erros são documentadas, analisados e comparados a erros de sistema, operação no caso de falta de treinamento.

Das especificações dos requisitos do software:

- **Correção:** se o requisito no documento é um requisito que tenha que ser implementado pelo software;

- **Não ambiguidade:** apenas uma interpretação para cada requisito no documento;

- **Completitude:** Se estão inclusos todos requisitos significantes, as definições das respostas do software a todos os tipos de dados em todas as situações possíveis, válidas ou inválidas;

- **Consistência:** se não há conflitos de um requisito com outro em um mesmo documento;

- **Classificação:** indicações no documento com relação a importancia do requisito;

- **Ser verificável:** para cada requisitos contido no documento, exista um processo finito e economicamente viável no qual uma pessoa ou maquina possa asegurar que o produto do software atenda o requisito;

- **Ser modificável:** a forma que as modificações serão atribuidas ao documento;

- **Ser rastreável:** se os requisitos são claros com relação a sua origem, e a facilidade da referência de cada requisitos nos documentos subsequentes do processo ou em melhoria da documentação do sistema.

Na identificação de requisitos Pfleeger (2004, p. 113) separa os requisitos em três categorias. “Requisitos que devem ser totalmente satisfeitos, altamente desejáveis, mas não necessários e possíveis, mas poderiam ser eliminados”

1.1.2 Requisitos funcionais

Requisitos funcionais de acordo com Sommerville (2007, p.81) “descrevem o que o sistema deve fazer, esses requisitos dependem do tipo de software que está sendo desenvolvido”

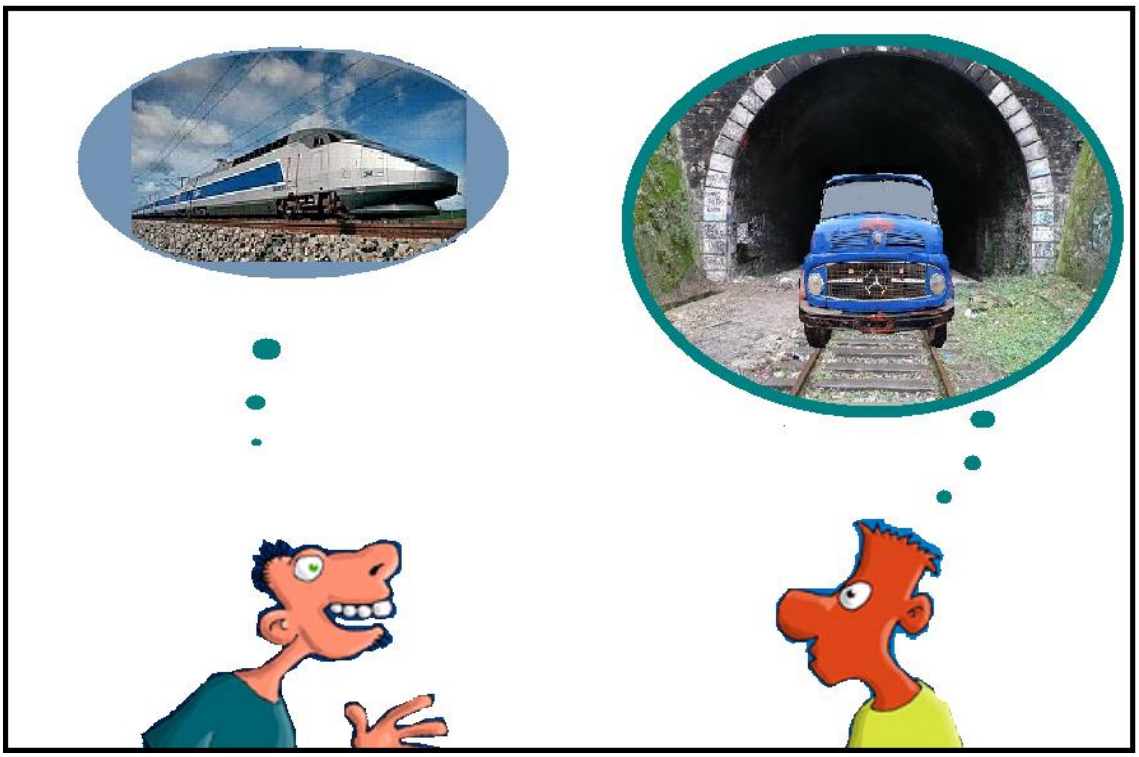
Os requisitos funcionais são direcionados a que o sistema irá fazer, que dependem do tipo de software que será desenvolvido, do tipo de usuário que o software esta sendo destinado e da abordagem geral redigido pela organização. Requisito funcional descreve as relações entre o sistema e o ambiente. Por exemplo os requisitos funcionais de um sistema de controle de estoque:

1. Relatórios curva ABC de produtos por preço;
2. Lista de produtos vendidos por data;
3. Relatórios de custo do produto.

Essas opções são requisitos de definição do usuário, o analista passa esses requisitos em uma linguagem com restrições, assim os riscos de erros na programação são mínimos. Considerando que o analista é um interprete, que entende tanto a maneira

do usuário pensar como o programador, tornando os termos abstratos compreesiveis e organizando em divisões lógicas. Sem a análise de requisitos, seria dificultoso o caminho dos programadores, quanto a forma de interpretação.

Figura 3 – Visão do programador sem o Analista



Gosseiramente imaginamos que requisitos são funções que serão incorporadas no software, dos documentos que definirão e especificarão os requisitos, incluem os seguintes tipos de itens: Ambiente físico, interfaces, os usuários, funcionalidade, documentação, dados, recursos, segurança, garantia de qualidade. Além das especificações dos requisitos do Software: correção, não ambiguidade, completitude, consistencia, classificação, ser verificável, ser modificável e ser rastreável.

1.1.3 Requisitos não Funcionais

Requisitos não funcionais de acordo com Sommerville (2007, p.82) “são aqueles não diretamente relacionados as funções especificas fornecidas pelo sistema”.

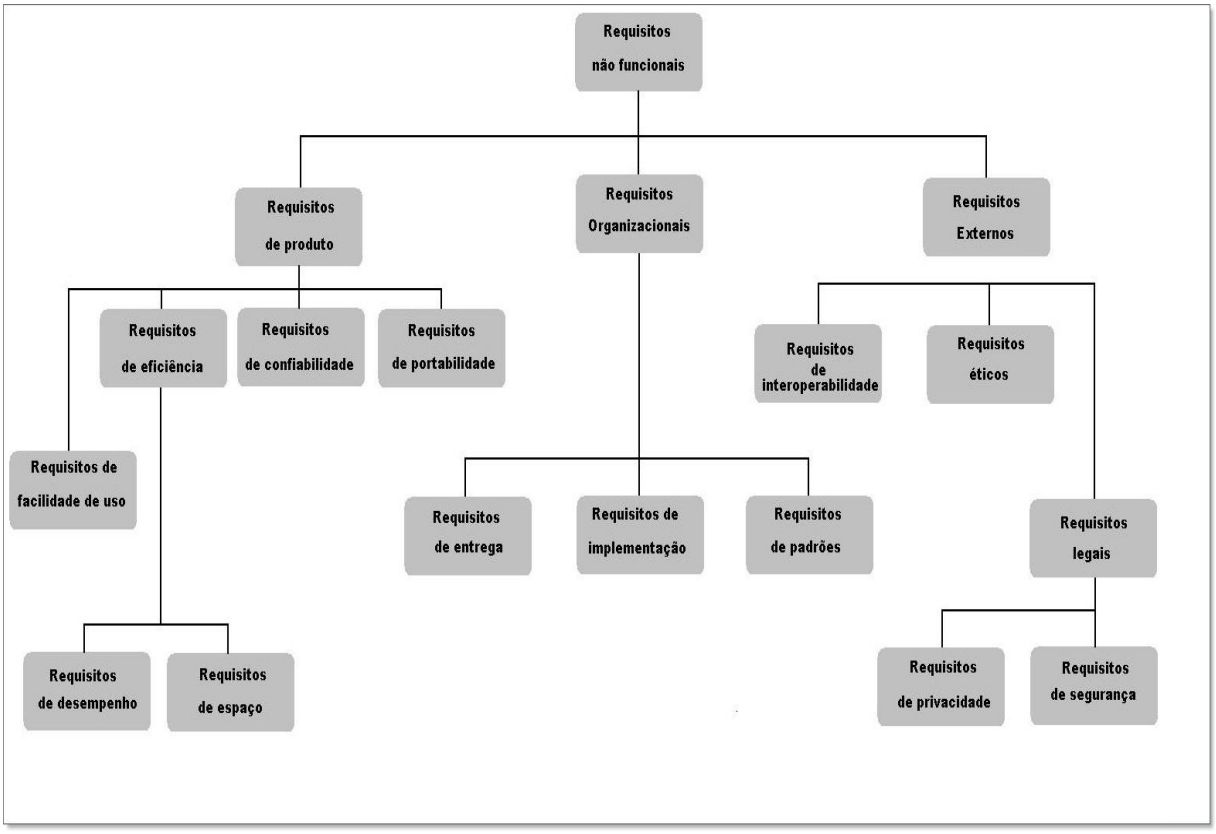
Os Requisitos não funcionais descrevem atributos relacionados ao ambiente,

diretamente relacionados às funções especificas fornecidas pelo sistema. Os requisitos descrevem o que o sistema fará, e a parte que se preocupa com o ambiente em que o sistema irá trabalhar, é chamado de requisitos não-funcionais ou restrições. É a forma de prever a compatibilidade do ambiente fisico em que o cliente esta trabalhando com a capacidade desenvolvida, de forma que, não se desenvolva um sistema complexo sem a preocupação com os equipamentos que o cliente possui. Levando em consideração a capacidade dos dispositivos de E/S (entrada e saída), espaço de armazenamento e tempo de resposta.

Os requisitos não-funcionais não somente estão relacionado ao sistema e suas funcionalidades, mas também estão ligados a fatores externos apresentados na Figura 4, como regulamentos de segurança e legislação. De acordo com Sommerville (2007, p. 83) os requisitos não funcionais são divididos em três partes.

- Requisitos de produto, que especificam o comportamento do sistema;
- Requisitos organizacionais, derivados de politicas e procedimentos da organização
- Requisitos externos, derivados de todos os fatores externos ao sistema e seu processo de desenvolvimento.

Figura 4 – Requisito Não Funcional



Fonte: Sommerville

1.2 Projeto de Software

O projeto é a fase de transformar o problema em solução, momento de tomar decisões lógicas sobre o software. Partindo dos requisitos e traduzindo o desejado pelo cliente em solução. Na fase de Projeto são determinados os aspectos pelo quais o sistema será montado, ou seja, com base na análise de requisitos são escolhidas as ferramentas que serão utilizadas para o densenvolvimento, o tipo de aplicação, a estrutura de dados. Esta fase não chega a ser técnica mas é onde se modela o que será utilizado na codificação.

Há uma distância que difere os requisitos de um projeto; para explicar damos o exemplo de uma criança que deseja uma festa de aniversário. A criança pede:

- Jogos para brincar com os amigos
- Vários tipos de doce
- Bolo com o simbolo do seu time de futebol preferido

Os pais anotam os desejos (requisitos) e projeta uma festa de acordo com o pedido da criança. Por exemplo, jogos para brincar com os amigos. Na imaginação da criança esses podem ser eletronicos, porém nos pensamentos dos pais existem varias opções de jogos além dos eletronicos. A maioria das opções de jogos analisadas pelos pais atenderiam os desejos da criança, porém existe como atingir o desejado pela criança, basta os requisitos serem bem preparados que o projeto atenderá perfeitamente o imaginado pela pequena criança.

No projeto de software a funcionalidade de atingir um determinado objetivo de acordo com a análise de requisitos, funciona de maneira semelhante ao exemplo dado. Após identificar uma solução para determinado problema deve-se levar em consideração se essa solução satisfaz as todos os requisitos especificados,

Pressman (1995, p. 130), em seu livro sobre engenharia de software, apresenta a seguinte definição:

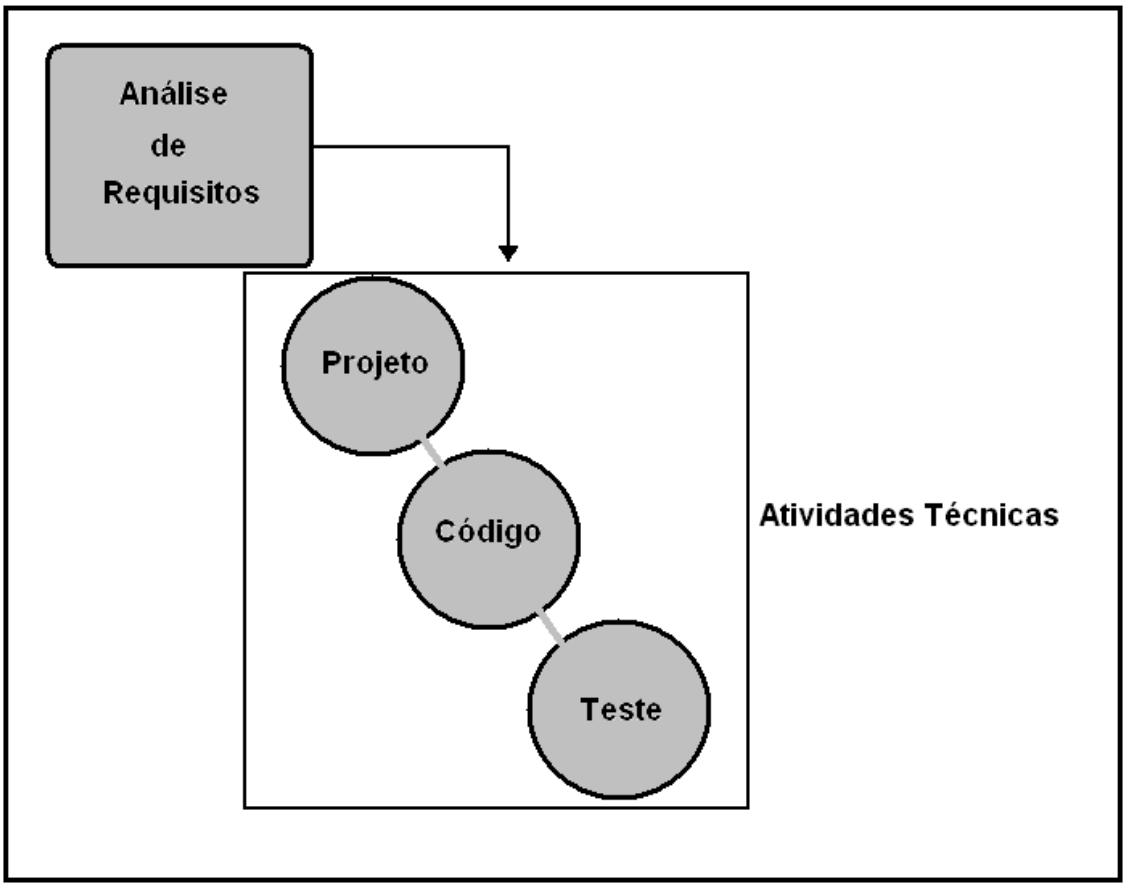
> Após compreender os requisitos funcionais do software, características de desempenho, restrições relativas ao sistema e preocupações de confiabilidade, o planejador aplica técnicas e ferramentas para deduzir estimativas de esforço e de tempo. Mas essas estimativas oferecem-nos informações úteis somente se estiverem integradas numa estrutura de planejamento mais completa.

O Projeto nada mais é que a documentação do sistema, que será utilizada posteriomente para comparação com os resultados obtidos.

1.2.1 O Porque do Projetista

Projeto de software é o primeira das três atividades técnicas – *projeto, codificação e teste* – necessárias para contruir e verificar um software, mostrado na figura 5. A importancia do projeto de software vem com a necessidade de uma exigencia quase unica no processo de construção de software – *qualidade*. O projeto de software serve como base para todos os passos de engenharia e manutenção de software, sem projeto é arriscado construir um sistema, sendo sua contrução instável e os resultados improváveis.

Figura 5 – As Três Atividades Técnicas



1.3 Implementação

Após analisar cuidadosamente os requisitos, entendendo o desejo do cliente/usuário e decidindo como o projeto solucionará os problemas encontrados e após de criar uma solução de alto nível para os mesmo, vem a próxima fase; a fase de implementação, onde deve-se concentrar em escrever programas que implementam o projeto. A considerar nesta fase, como um sistema afetará as práticas de trabalho, como mudará o sistema de trabalho, como interagirá com outros sistemas e quais as mudanças significativas na maneira de operação do usuário e quais devem ser automatizadas. Ou seja, na implentação, tudo que foi determinado anteriormente começa a ser modelado no banco de dados, codificado em uma linguagem de programação, compilado, entre outras rotinas técnicas, dependendo das ferramentas de desenvolvimento que estão sendo utilizadas.

No modelo de desenvolvimento cascata, essa fase irá gerar o software para ser executado, este deverá atingir os objetivos determinados na fase Análise de Requisitos.

CAPÍTULO 2 – FASE DE TESTE

A fase de teste consiste em encontrar erros ou futuros problemas que possam prejudicar o funcionamento desejado de um software. Trata-se de uma tarefa da engenharia de software aplicada após a construção de um software, que parte de um conceito abstrato para algo tangível.

> O engenheiro cria uma série de casos de teste que tem a intenção de "demolir" o software que ele construiu. De fato , a atividade de teste é um passo do processo de engenharia de software que poderia ser visto (psicologicamente, pelo menos) como destrutivo, em vez de construtivo (PRESSMAN, 1995 p. 787).

Encontrar erros em sistemas complexos é uma tarefa muito complicada, podendo levar muito tempo para testar todos os componentes e processos do sistema.

Algumas técnicas podem facilitar a tarefa do testador, sendo um método que inicia, nas unidades do sistema, logo após uni essas unidades de sistemas, método chamado teste de integração, e por fim testar todas as unidades chamadas teste de sistema, com objetivos de ter a certeza de que o software desenvolvido é totalmente confiável. Testar software é encontrar erros (bugs) que são erros no funcionamento de um software, também chamado de falha de lógica.

O termo bug é freqüentemente utilizado para referir-se a um problema ou falha em um computador. O termo teve origem nos Estados Unidos, no momento em que os pioneiros de computadores construíam sem ser de válvulas, quando havia uma série de falhas inexplicáveis, acabaram por ser rastreadas e eram nada além de mariposas voando dentro do computador.

Teste de software não deve ser confundido com a depuração. Depuração é o processo de analisar e localizar bugs de software quando não se comporta como o esperado. Embora a identificação de alguns bugs serão evidentes a partir de brincadeiras com o software, uma abordagem metódica para teste de software é muito mais profundo um meio de identificar erros. Depuração é, portanto, uma atividade que apóia testes, mas não podem substituir os testes. No entanto, nenhuma quantidade de testes pode ser garantida a descobrir todos os bugs.

> Há um mito segundo o qual, se fossemos realmente bons para programar não haveria bugs a ser procurados.Se pudéssemos realmente nos concentrar, se todos usassem programação estruturada, projeto, top-dow, tabelas de decisão, se os programas fossem escritos em SQUISH, se tivéssemos as balas d prata certas, então não haveria bugs. Assim segue o mito. Existem bugs, diz o mito, porque somos ruins nquilo que fazemos; e, se somos ruins nisso, devemos sentir-nos culpados por isso. Por conseguinte, a atividade de teste e o projeto de casos de teste são uma admissão de falha, o que promove uma boa dose de culpa. E o tédio de testar é apenas uma punição por nossos erros. Punição por quê ? Por sermos humanos? Culpa por quê? Por deixarmos de conseguir uma perfeição sobre-humana? Por não distinguirmos entre o que outro programador pensa e o que ele diz? Por deixarmos de ser telepáticos? Por não resolvermos os problemas de comunicação humana que têm existido por aí... 40 séculos? (BEIZER 1990, p.01 apud PRESSMAN,1995,p.787).

São vários os tipos de métodos que a engenharia de software utiliza para a construção de software. Para explicarmos as principais fases de um desenvolvimento de software, estamos considerando o modelo cascata, porém, independente dos métodos de desenvolvimento, o teste esta nas diversas formas de construção de software. De acordo com as evoluções nos sistemas de desenvolvimento, a fase de teste aumenta sua importância sendo mais exigida.

2.1 Fases de Teste de Software

Analisando os métodos e processos atuais, encontramos objetivos em comum. Assegurar que todos os requisitos deverão ser testados, especializar uma equipe e um encarregado, ser um método formal.

Na década de 80 uma variedade de métodos de projeto de casos de teste evoluiu. Esses métodos oferecem ao desenvolvedor uma abordagem sistemática ao teste. Porem não destacava certos assuntos e não davam a ênfase necessária deixando a equipe passar despercebido nos detalhes como:

- Importância dos processos de testes;
- Defeitos só detectados em produção custam de 100 a 1000 vezes mais para serem consertados;
- Abordagens únicas para novas tecnologias;
- Falta de automação para testes;
- Falta de metodologia para testes.

2.2 Teste de Unidade

O teste de unidade concentra o esforço do testador na menor unidade do software (componente ou módulo de software) sendo possível testar de maneira eficaz todo sistema, porém divido em partes.Essa técnica utiliza a descrição do projeto em nível de componente como guia, e todos os caminhos de controle são testados para descobrir defeitos dentro dos módulos.

Pressman (1995 p. 787):

> A Atividade de teste de unidade normalmente é considerada um adjunto da estapa de codificação. Depois que o código-fonte foi desenvolvido, revisado e verificado a fim de conseguir uma sintaxe correta, inicia-se o projeto do caso de teste de unidade.

O teste de unidade tem como base sempre a caixa-branca, método realizado em paralelo para múltiplos módulos.

2.3 Teste de Integração

Teste de integração consiste em uma técnica sistemática para o processo de junção de todos os componentes desenvolvidos, é uma técnica para descobrir erros associados as interfaces do software. O teste de integração é utilizado na fase de teste para verificar a funcionalidade do software em conjunto, onde juntando todos os componentes se possibilita visualizar onde esta o erro. Existem dois métodos de teste de integração:

- Integração Top-Down;
- Integração Bottom-Up;

Que depende das características do software para escolha de qual método mais eficiente para testar.

2.3.1 Integração Top-Dow

Na estrutura geral de um sistema, primeiro é criado um esqueleto geral, o programa principal. Os módulos ou (componentes) são adicionados ao esqueleto esse método é chamado teste de integração Top-Down .

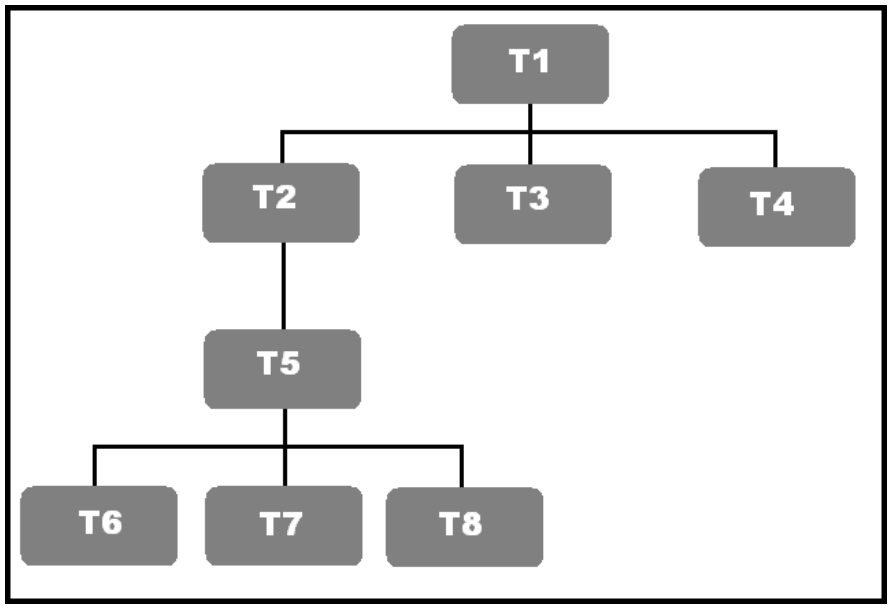
Integração top-down inicia-se do programa principal até os módulos periféricos (módulos subordinados). O método de teste de integração Top-Down aborda à construção da estrutura do programa movimentando em um critério de prioridade de cima para baixo, ou seja, do modulo mais importante chamado (programa principal) até os módulos subordinados. Os módulos são adicionados ao esqueleto de duas formas.

- Depth-First, de acordo a profundidade;
- Breadth-First de acordo com a largura.

Em exemplo, mostramos na figura 6, o modelo de teste estrutural top-down, sendo os módulos adicionados de forma depft-first. Sendo T1 adicionado primeiro, como segunda opção o modulo T2. De acordo com o sistema Depth-First o próximo modulo adicionado é o T5, com o T6 em sequência.

No método Bradth-first, o adicionamento dos componentes começa no componente T1, seguido do modulo T2 sendo o próximo modulo o T3 seguido do T4.

Figura 6 – Os Componentes no Modelo Estrutural Top-Down



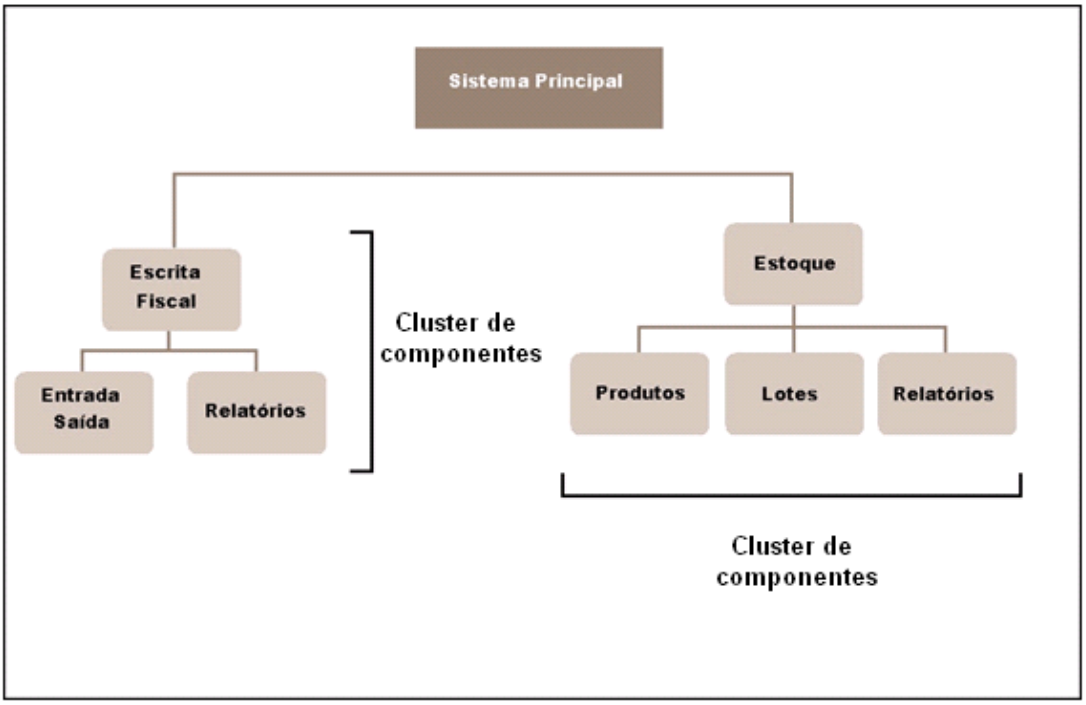
Independente do método ser Depth-first ou breadth-first o testes são realizados de acordo a adição dos módulos.

2.3.2 Integração Bottom-Up

Como o nome propõe, o teste de integração bottom-up inicia dos componentes (módulos) localizados no nível mais baixo até ao topo dos sistema. Sendo os componentes de infra-estrutura com acesso à rede e ao banco de dados semelhantes ou seja componentes que fornecem serviços em comum. Em outras palavras é uma maneira de testar as partes menos complexas de um sistema adicionando ao sistema tornando o teste cada vez mais complexo.

Os módulos de baixo nível são unidos formando um cluster de componentes executando funções especificas em um sistema. Exemplo Figura 7.

Figura 7 – Representação do Cluster no Modelo de Integração Bottom-Up



2.4 Teste de Sistema

Teste de sistemas é um conjunto de diferentes tipos de testes com um mesmo objetivo, funcionar todas as partes dos software. É o momento de por a prova todo o software e analisar sua funcionalidade, se todos os elementos do sistema estão integrados adequadamente e se os sistema esta de acordo com as funções atribuídas.

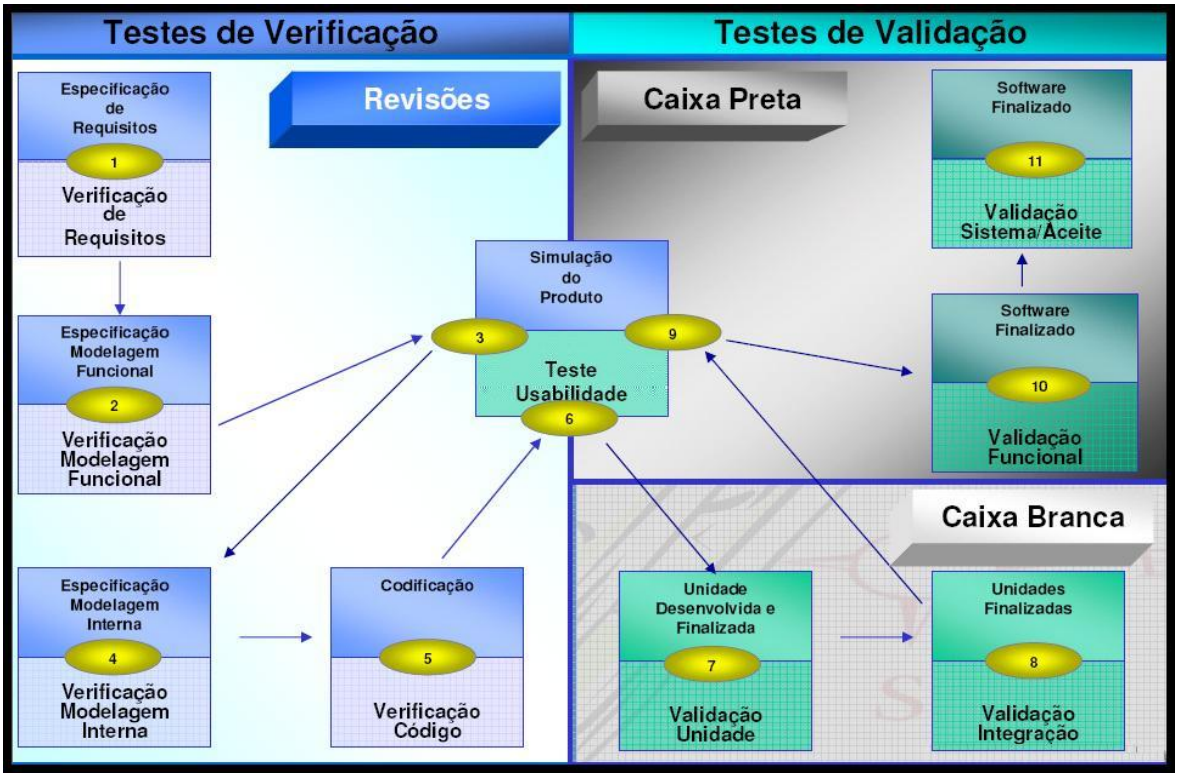
Sommerville (2007, p. 357):

> O teste de sistema envolve a integração de dois ou mais componentes que implementam funções ou características de sistema e depois o teste desse sistema integrado. Em um processo de desenvolvimento iterativo, o teste de sistema concentra-se no teste de um incremento que será entregue ao cliente: em um processo em cascata, o teste de sistema concentra-se no teste de todo o sistema.

2.5 Teste de Verificação e Teste de Validação

Teste de software é o processo de execução de software de uma maneira controlada. É freqüentemente usada em associação com os termos verificação e validação. A verificação é o ensaio de itens, incluídos nos software, para a conformidade e a consistência com uma especificação associada. Teste de software é apenas um tipo de verificação, que também usa técnicas tais como estudos, análises e inspeções. A validação é o processo de verificação de que, o que foi especificado é o que o usuário realmente deseja.

Figura 8 – Tipos de testes de Verificação e Validação



Fonte: Andreas Spillner

2.6 Técnicas de Teste de Software

Assegurar a satisfação do cliente e proporcionar ao cliente segurança e confiabilidade. Para obter esses fatores hoje cada vez mais exigidos, no mercado, o processo de desenvolvimento de software deve se preocupar muito com a fase de teste.

Para testar software existem algumas técnicas que contribui para eficácia da atividade de teste, neste capitulo vamos estudar as técnicas de teste funcional, estrutural, e baseada em erros.

2.6.1 Teste Funcional

O programador terminando de escrever todo o sistema, da inicio ao primeiro teste de sistema, o teste funcional, também conhecido como “black Box” (caixa preta) onde foca o comportamento externo do sistema, ignorando a estrutura do sistema e enfatizando a funcionalidade, no caso não se sabe se o esperado erro é problema de programação ou uma lógica estruturada.

Concentra-se nos requisitos funcionais do software, se eles estão satisfazendo as necessidades pré-estabelecidas, com a visão do usuário. Testando a funcionalidade do sistemas podemos encontrar os seguintes tipos de falhas:

- Funções incorretas
- Funções não implementadas
- Erros de Interfaces
- Erros de desempenho
- Erros de inicialização e finalização

2.6.2 Teste Estrutural

Teste estrutural ao contrário do teste funcional que foca o comportamento externo do sistema, o estrutural tem como finalidade abordar o funcionamento interno do sistema, Sommerville (2007, p. 367) reconhece como “teste caixa-branca, teste caixa-de-vidro ou teste caixa-clara”.

Na utilização do método caixa branca, a equipe de teste de software garante que todos os caminhos sejam exercitados pelo menos uma vez, todas as decisões lógicas do software seja executada, todos os laços e estruturas de dados internas.

2.6.3 Teste Baseado em Erros

Teste baseado em defeitos analisa no desenvolvimento de software, informações sobre os diversos tipos de defeitos, ou seja os que ocorrem com mais freqüência. Focando erros que o programador pode cometer no processo de desenvolvimento do software.

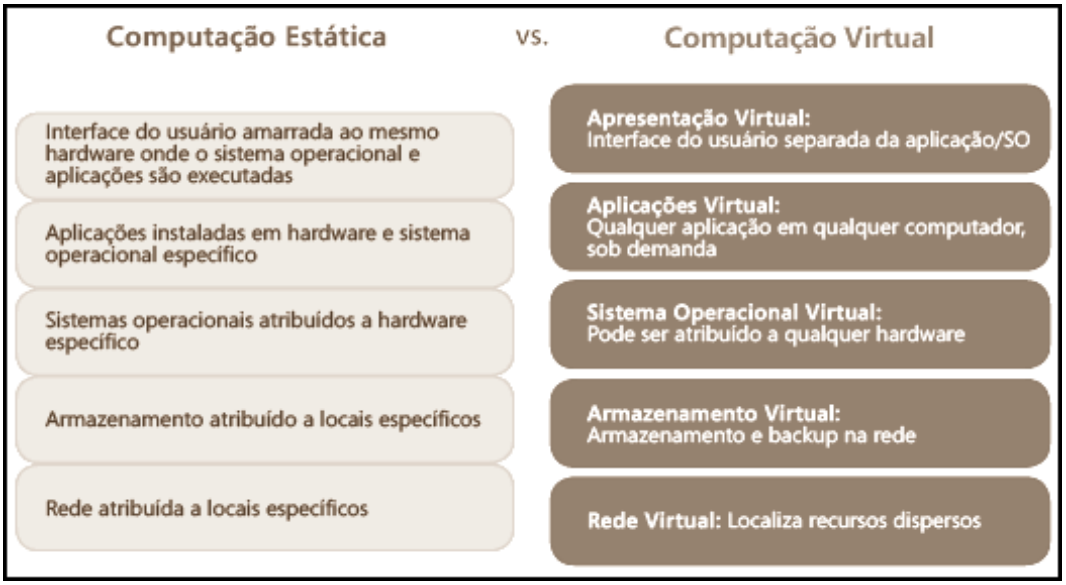
Lawrence(2004, p271):

> Quando não existe nenhum defeito óbvio, testamos nossos programas para verificar se podemos isolar mais defeitos criando condições em que o código não reage como planejado. Assim, é importante saber que tipo de defeitos estamos procurando.

2.7 Virtualização como apoio de Teste de Seftware

A virtualização começou em 1998 com a VMware. A virtualização possibilita a relação direta entre as partes das camadas de sistema operacional ao hardware, a aplicação ao sistema operacional e a interface do usuário à máquina local. Vemos na figura 9 uma comparação entre o modo de computação estática e o modo de computação virtual. No modo Virtual o sistema operacional é executado usando Virtualização de hardware, diferente do modo convencional que necessita de hardware especifico.

Figura 9 – Comparação entre as camadas Estáticas e VIrtual



Fonte: Microsoft

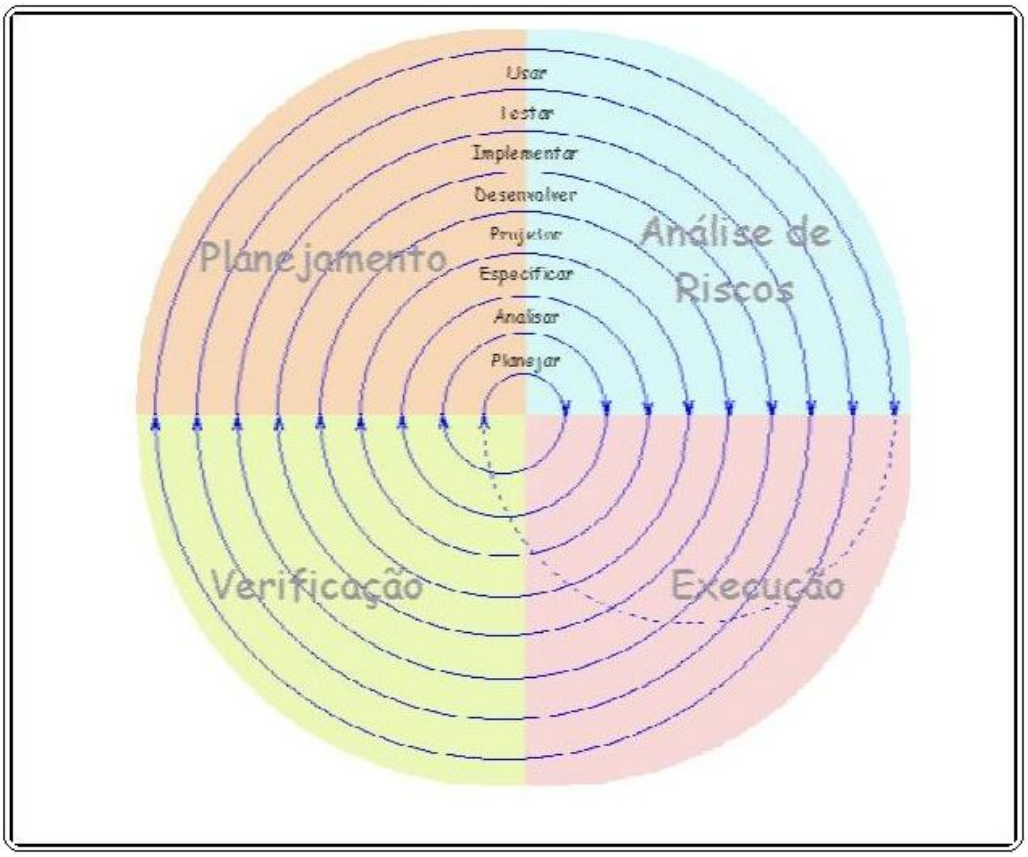
Mas como a virtualização pode ser usada como uma ferramenta de teste?
Como a virtualização possibilita em uma mesma máquina ter vários sistemas operacionais, é possível utilizar a virtualização para testar funcionalidade do software em plataformas diferente, testando o comportamento do software em sistemas operacionais diferente, e diminuindo a capacidade de memória da maquina, visualizar o comportamento do software em situações de trabalho não favoráveis ao usuário.

2.8 Teste nos Modelos de Desenvolvimento

2.8.1 Teste no modelo de desenvolvimento espiral

O **modelo em espiral** desenvolvido em 1988 por Boehm, tendo como tentativa integrar os diversos modelos existentes da época, eliminando suas dificuldades e explorando seus pontos fortes. Sendo a fase teste revisada de acordo com a evolução da construção do sistema. Um dos métodos que mais utiliza a equipe de teste.

Figura 10 – Modelo de Desenvolvimento Espiral:

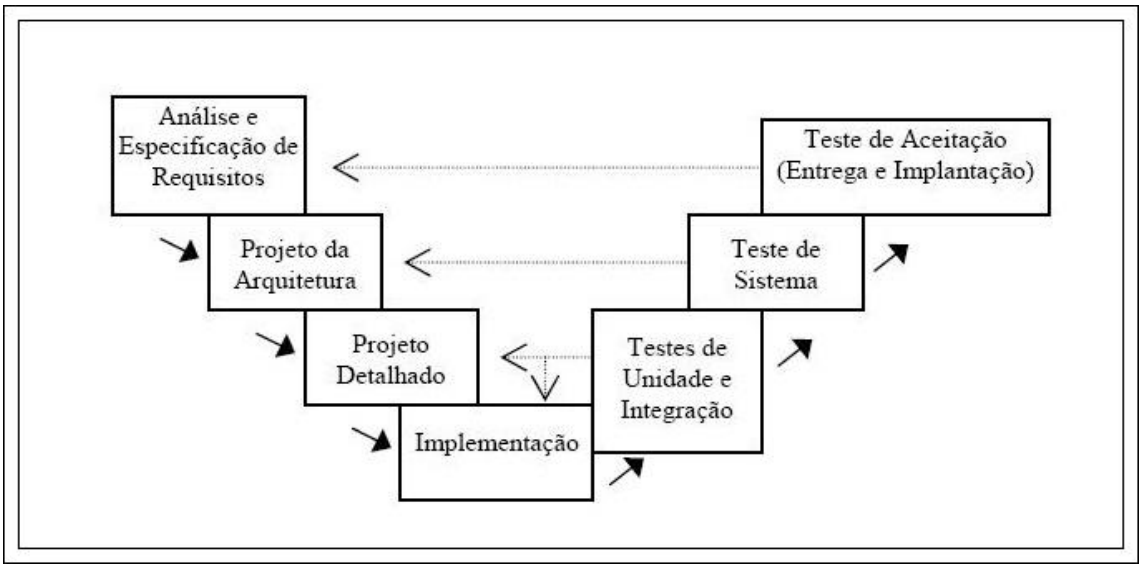


Fonte: Sommerville

2.8.2 Teste no modelo de desenvolvimento em V

Os testes em modelo em V criados em 1992, são utilizados basicamente para verificar a implementação. Sendo os testes de integração focados na integração das unidades que compõem o software, usados para avaliar todo o projeto nos seus detalhes. Assim os testes de unidades e integração devem garantir que todos os aspectos do projeto do sistema foram implementados corretamente no código. Quando os testes de integração atingem o nível do sistema como um todo (teste de sistema), o projeto da arquitetura passa a ser o foco. Neste momento, busca-se verificar se o sistema atende aos requisitos definidos na especificação. Finalmente, os testes de aceitação, conduzidos tipicamente pelos usuários e clientes, valida os requisitos, confirmando que os requisitos corretos foram implementados no sistema (teste de validação).

Figura 11 – Modelo de desenvolvimento em V



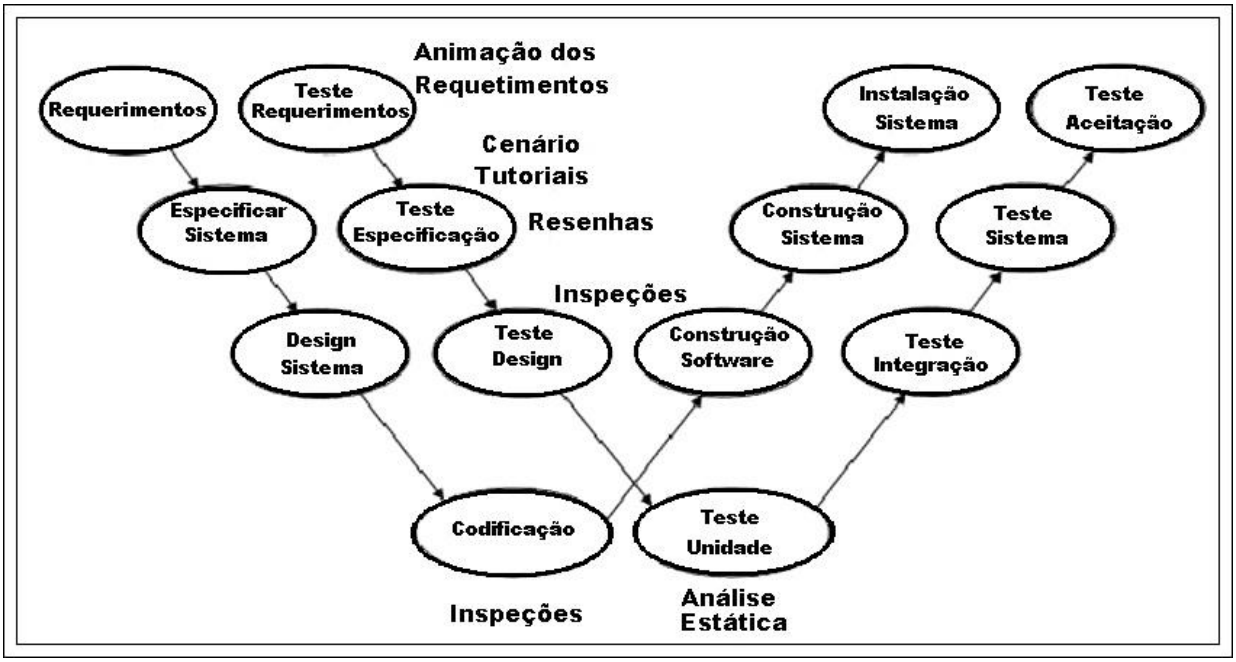
Fonte: Andreas Spillner

2.8.3 Teste no modelo de desenvolvimento em W

O Model-W, desenvolvido na tentativa de resolver deficiências apresentadas no modelo em V. Em vez de incidir sobre determinadas fases de ensaio dinâmico, como o modelo V faz, o modelo em W enfoca o desenvolvimento dos produtos próprios, essencialmente, o desenvolvimento de cada atividade. Com objetivo de determinar no desenvolvimento, se esta de acordo com as exigências. Na sua forma mais genérica, o modelo em W, apresenta um padrão de desenvolvimento *lifecycle*. Onde cada etapa desenvolvida é feito uma atividade de teste. No lado esquerdo, as atividades são tipicamente de desenvolvimento (por exemplo, escrever requisitos) junto com atividade de teste dos requisitos e assim por diante.

Sua organização tem um conjunto em diferente fases de desenvolvimento, o modelo em W é facilmente adaptado à sua situação, centra-se especificamente sobre os riscos do produto no local onde a preocupação teste pode ser mais eficaz.

Figura 12 – Modelo de Desenvolvimento em W



Fonte: Andreas Spillner

Testes de aceitação podem fazer uso da concepção funcional, o modelo em W é bastante artificial elimina o constrangimento de terem o mesmo número de etapas de desenvolvimento com teste dinâmico. Se existem cinco fases de desenvolvimento preocupado com a definição, concepção e construção de código em seu projeto, talvez fosse sensato ter apenas três fases de testes dinâmicos.

2.9 A fase de Teste para diminuir a Manutenção

O foco da fase de testes é descobrir erros, problemas que da implantação do sistema desenvolvido ao longo da utilização ocorrerá, sendo necessário a fase de manutenção.

Pressman (1995, p. 881), fala sobre a manutenação de software:

> Na década de 1970 a manutenção era responsável por um índice entre 35% e 40% do orçamento de software para uma organizção de sistemas de informação esse valor pulou para aproximadamente 60% durante a década de 1980.

O teste tem sido a solução para evitar constragimentos entre a empresa desenvolvedora e o cliente. Teste é atualmente responsável pela maior porcentagem de esforço técnico no processo de desenvolvimento de software.

A fase de teste sendo responsável pela diminuição da manutenção, tornou-se ponto-chave no desenvolvimento de software, sendo necessário o investimento e toda a atenção, criação de equipes de teste e de metodos para testar software.

CAPÍTULO 3 – QUALIDADE DE SOFTWARE

A Qualidade de Software é definida como realização de medições no desenvolvimento de software para obter cada vez mais qualidade no projeto, no processo e produto, na busca constante da satisfação plena dos clientes, dos usuários e custo compatível com o produto oferecido.

3.1 A Origem e Evolução da Qualidade

Antigamente as pessoas produziam, faziam as inspeções e vendiam os produtos. Essas pessoas eram chamadas “artesãos”, tinham contato direto com os clientes, sabiam as especificações, os requisitos e satisfação que os clientes esperavam do seu produto.

Com a evolução do comércio começaram então a surgir os intermediários, os produtores não tinham mais contato direto com seus clientes e não podiam saber com clareza sobre as expectativas que os clientes tinham dos produtos, os artesões então tiveram que realizar a inspeção da qualidade do seu trabalho.

No final do século XIX , com a revolução industrial a produção passou a ser em massa, e a partir desse momento surgir o cargo de supervisor de produção que era responsável por coordenar as pessoas e desenvolver as tarefas. O supervisor passou a ser responsável pelo controle de qualidade dos produtos.

Com a Primeira Guerra Mundial foram criadas as primeiras funções segregadas com tarefas especificas para realização de inspeções de qualidade passaram a ser dessa função.

Na Segunda Guerra Mundial a produção passou a ser em larga escala de material bélico, foram colocados inspetores como primeira ferramenta de estatística de controle de qualidade. A realização de inspeções com base nas técnicas de amostragem a inspeção era feita em cem por cento dos produtos e controle da conformidades dos produtos através de gráficos.

A técnica estatística da qualidade teve grande desenvolvimento, sendo responsável por parte do bem-sucedido da guerra dos norte-americanos.

A aplicação das técnicas de controle estatísticos da qualidade foi inserida no período pós-guerra no Japão, pelas as forças americanas, visando auxiliar na reconstrução do pais, foi colocado na década 50 o controle de qualidade estático.

Na década 60 com início da globalização da economia o aumento do nível de exigência por produtos com mais qualidade, introduziu o Controle da Qualidade Total (CQT), essa idéia foi desenvolvida pelos japoneses. O CQT foi a base da competição mundial.

A idéia de CQT, métodos e práticas na década 80 evoluíram para Controle da Qualidade por todas as empresas, abrangendo todas as funções que empresa, tanto os fornecedores e distribuidores a fim de garantir a qualidade dos produtos e serviços conforme requisitos que os consumidores buscam, bem como a Gestão da Qualidade Total.

Só na década de 80 que os ocidentais descobriram o motivo do sucesso japonês, mais tarde no ano de 1987 foram criadas as primeiras normas internacionais sobre o Sistema da Qualidade, as normas ISO 9000.

3.2 Conceito da Qualidade

Feigenbaum (1986) apud Fernandes (1995, p. 28), fala sobre o conceito de qualidade:

> Qualidade é a composição total das características de marketing, engenharia, fabricação e manutenção de um produto ou serviço, através das quais o mesmo produto ou serviço, em uso atendera as expectativas do cliente.

Juram (1988) apud Fernandes (1995, p. 28), define: “Qualidade é a adequação ao uso do ponto de vista do cliente.”

De acordo com America Society for Quality Control (1983) apud Fernandes (1995, p. 28): “Qualidade é a totalidade de requisitos e características de um produto ou serviços que estabelecem a sua capacidade de satisfazer necessidades implícitas e explicitas.”

Ishikawa (1993) apud Fernandes (1995, p. 28), cita o conceito qualidade:

> Interpretado de forma mais restrita, qualidade significa qualidade do produto. Interpretado de forma mais ampla, qualidade significa qualidade de trabalho, qualidade de serviço, qualidade de informação,

> qualidade de processo, qualidade de sistema, qualidade de empresa, qualidade de objetivos.

De acordo com Falconi (1992) apud Fernandes (1995, p. 28), cita sobre o conceito qualidade: “Um produto ou serviço de qualidade é aquele que atende perfeitamente, de forma confiável, de forma acessível, de forma segura e no tempo certo, as necessidades do cliente.”

De acordo com Crosby (1984) apud Fernandes (1995, p. 28): “Qualidade é conformidade com os requisitos.”

Apesar de não haver uma única definição, podemos definir que a qualidade é um termo de necessidade e expectativas que do cliente, nos requisitos dos produtos e serviços dos projetos, os processos de fabricação, marketing, entrega, instalação e assistência técnicas, esses requisitos evoluíram com tempo, por isso nenhuma empresa pode chegar ao topo da qualidade.

3.3 Certificação

Modo pelo qual dá garantia escrita de que um produto, processo ou serviço está em conformidade com os requisitos especificados.

3.3.1 Certificação de Software

Para emissão de um certificado de conformidade de um software há um certo conjunto de normas ou especificações, comprovando os testes de efetuados por certificadoras.

Segundo a norma ISO 9000:2000, qualidade é o grau em que um conjunto de características inerentes (a um produto, processo ou sistema) cumpre os requisitos (daquele produto, processo ou sistema).

Segundo a norma ISO 9126, relacionada a produtos, qualidade é: "Um conjunto de atributos que têm impacto na capacidade do software de manter o seu nível de desempenho dentro de condições estabelecidas por um dado período de tempo”.

Alguns autores dizem basicamente que qualidade é estar em conformidade com os requisitos.

3.4 Características da Qualidade - NBR ISO/IEC 9126-1

Definição dos requisitos de qualidade de um produto de software:

Avaliação das especificações do software durante o desenvolvimento para verificar se os requisitos de qualidade estão sendo atendidos.

Descrição das características e atributos do software implementado por exemplo nos manuais de usuário.

Avaliação do software desenvolvido antes da entrega ao cliente.

Avaliação do software desenvolvido antes da aceitação pelo cliente.

Funcionalidade, satisfaz as necessidades, confiabilidade, numerar a falhas, usabilidade, fácil de usar, eficiência, rápido – Enxuto, Manutenabilidade, Fácil de modificar destina o produto.

- Funcionalidade: Evidencia que o conjunto de funções atenda às necessidades explícitas e implícitas para a finalidade a que se destina o produto.

- Confiabilidade: Evidencia que o desempenho se mantém ao longo do tempo e em condições estabelecidas.

- Usabilidade: Evidencia a facilidade para a utilização do software.

- Eficiência: Evidencia que os recursos e os tempos envolvidos são compatíveis com o nível de desempenho requerido para o produto.

- Manutenibilidade: Evidencia que há facilidade para correções, atualizações e alterações.

- Portabilidade: Evidencia a capacidade do software de ser transferido de um ambiente para outro.

3.5 O Teste de Software como Fator de Qualidade

Teste de Software é verificar se o software está fazendo o que deveria fazer, de acordo com seus requisitos, e não está fazendo o que não deveria fazer (Rios e Moreira – 2002).

Teste de Software é o processo que visa executar o software de forma controlada com o objetivo de avaliar seu comportamento, baseado no que foi especificado.

CAPÍTULO 4 – O MERCADO DE TESTE

Tendo em vista a maior busca por qualidade de software, e destacando-se a importância desta atividade, visualizam-se na prática algumas empresas empenhadas em desenvolver seus produtos e testá-los dentro dos métodos corretos, com profissionais capacitados para realizar o processo, que obtêm ferramentas e tecnologias adequadas para a realização dos testes.

Muitas empresas aproveitam-se dessa necessidade que os desenvolvedores de sistemas têm de testar seus produtos para se especializarem nesta atividade, e assim oferecer serviços de terceirização de Teste de Software sem que o próprio construtor do projeto tenha esse trabalho. Essas empresas podem ser chamadas de “Test Houses”. Nesse aspecto, o desenvolvedor não precisa formar um equipe de testes, adquirir estrutura física, obter ferramentas, ou disponibilizar de mais tempo para realização do processo de teste, basta apenas fornecer a empresa terceirizada os requisitos do projeto, o software a ser validado, e arcar com os custos desse serviço, para que assim os testes sejam realizados. É necessário que o gestor responsável da empresa de desenvolvimento obtenha corretamente as informações de qual forma será menos custosa, e a que trará mais benefícios ao software, para que a decisão de terceirização ou não do Teste de Software seja tomada da maneira mais eficaz.

Na prática, visualizamos a missão da empresa Quality Test, especialista em Teste de Software:

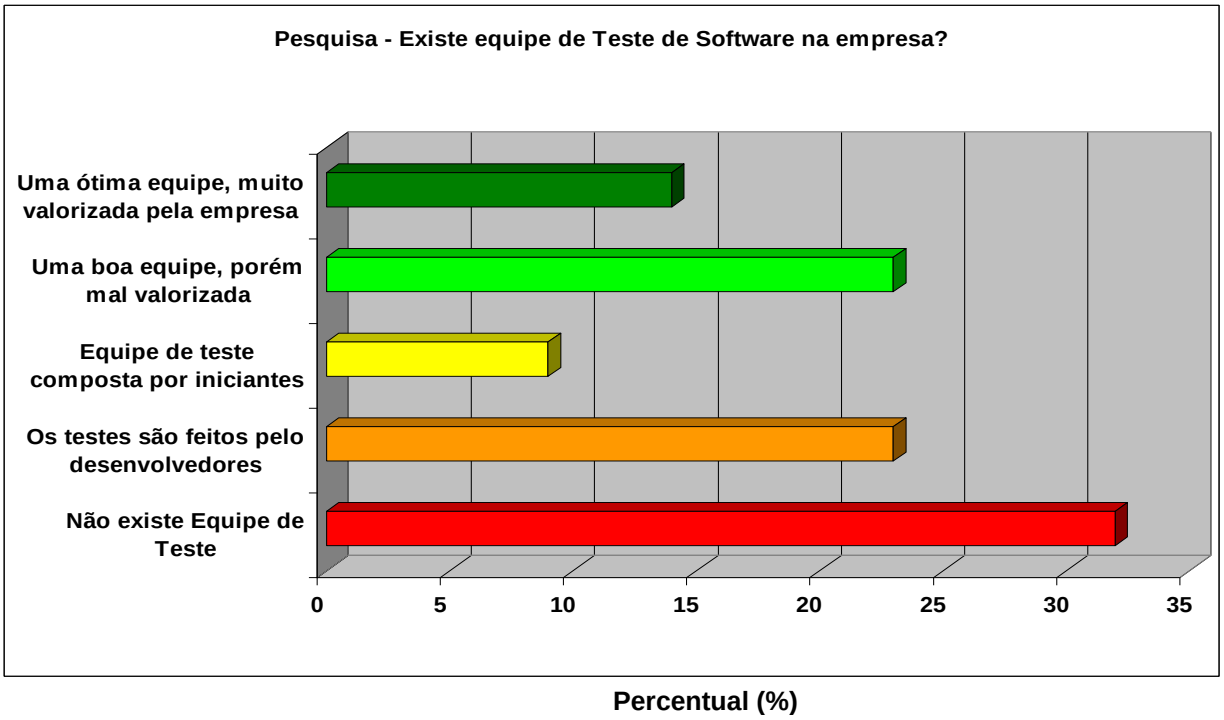
> Fornecer para as empresas de desenvolvimento de softwares, uma estrutura de teste seguindo padrões internacionais, estabelecendo relação com entidades conceituadas na área e garantindo assim uma melhor qualidade do software desenvolvido.

Apesar da evolução da atividade de teste no mercado, muitas empresas não realizam os testes de maneira adequada, e quando realizam, pouco valorizam este processo.

Uma pesquisa realizada neste trabalho com 43 profissionais da área de TI da região de Marília (estado de São Paulo) e em alguns outros municípios, mostra que 55% das empresas não realizam o processo de Teste de Software (32%) ou delegam essa atividade aos próprios programadores (23%), não aplicando os métodos adequados. A pesquisa mostra também que 45% das empresas possuem uma equipe exclusivamente

para o teste de software, mas na maioria delas a equipe é pouco valorizada (23%), podendo ser composta até por iniciantes ou estagiários (9%). Segue abaixo o gráfico que demonstra de maneira mais clara e detalhada a pesquisa realizada:

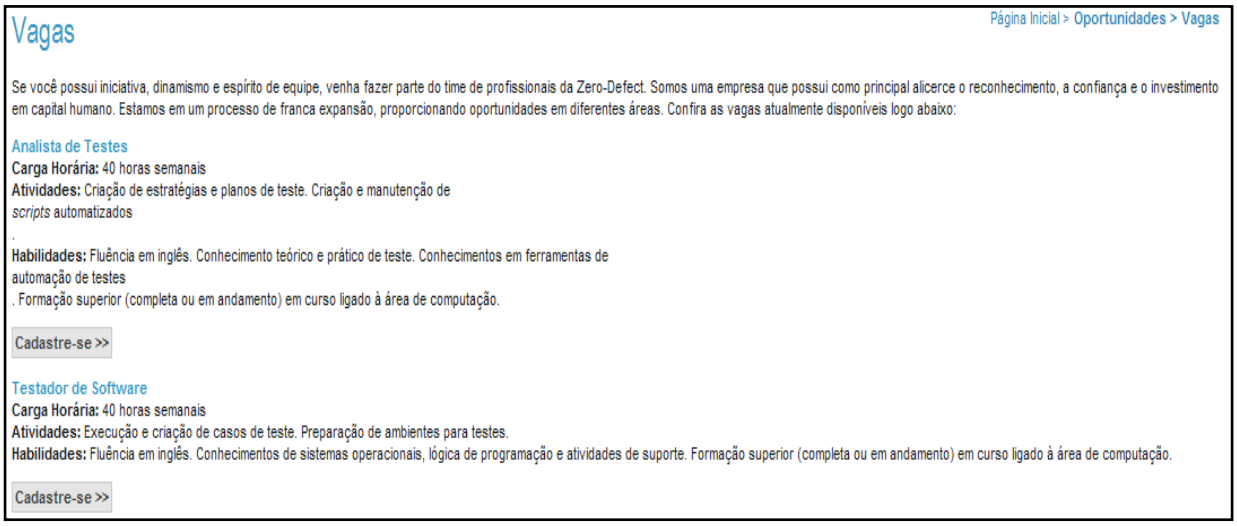
Gráfico 01 – Existência de Equipes de Teste nas Empresas



4.1 O Mercado de Trabalho

Devido o crescimento da área de Teste, profissionais capacitados hoje são muito procurados pelas empresas de desenvolvimento, e empresas especializadas em teste de software, as chamadas “Test Houses” (casas de teste), como mostra a figura seguinte retirada de um site de uma dessas empresas:

Figura 13 – Tablóide de Vagas na Área de Teste



Vagas

Página Inicial > Oportunidades > Vagas

Se você possui iniciativa, dinamismo e espírito de equipe, venha fazer parte do time de profissionais da Zero-Defect. Somos uma empresa que possui como principal alicerce o reconhecimento, a confiança e o investimento em capital humano. Estamos em um processo de franca expansão, proporcionando oportunidades em diferentes áreas. Confira as vagas atualmente disponíveis logo abaixo:

Analista de Testes
Carga Horária: 40 horas semanais
Atividades: Criação de estratégias e planos de teste. Criação e manutenção de *scripts* automatizados
.
Habilidades: Fluência em inglês. Conhecimento teórico e prático de teste. Conhecimentos em ferramentas de automação de testes
. Formação superior (completa ou em andamento) em curso ligado à área de computação.

Cadastre-se >>

Testador de Software
Carga Horária: 40 horas semanais
Atividades: Execução e criação de casos de teste. Preparação de ambientes para testes.
Habilidades: Fluência em inglês. Conhecimentos de sistemas operacionais, lógica de programação e atividades de suporte. Formação superior (completa ou em andamento) em curso ligado à área de computação.

Cadastre-se >>

Fonte: Empresa Zero Defect – 10/11/2008

Hoje no mercado de trabalho de tecnologia do Brasil, existe um problema de alta rotatividade de equipe de projetos, em diversas áreas, como banco de dados, programação, hardware, entre outras, isso ocorre devido às freqüentes propostas de trabalho que esses profissionais recebem de outras empresas, o que pode ser um risco para o projeto que estão ou foram desenvolvidos por esses colaboradores, já que é difícil a reposição de um profissional tão qualificado quanto o que foi perdido. Para prevenção desta situação, as grandes empresas procuram fornecer ao funcionário um plano de carreira, para que o mesmo possa se estabelecer na empresa sabendo que em um tempo determinado pela empresa ele irá atuar em um melhor cargo.

Com o crescimento da atividade de teste nas empresas e com a busca desses profissionais, hoje existem cursos de certificação na área, essas certificações tornam o profissional mais valorizado, devido a certeza de que as empresas têm que esse tipo de funcionário obtêm o conhecimento necessário para desempenhar a função desejada. Seguem abaixo algumas dessas certificações na área de Teste de Software:

- ALATS – Associação Latino-Americana de Teste de Software
  CBTS - Certificação Brasileira de Teste de Software

- ISTQB - International Software Testing Qualifications Board (ISEB)

Foundation Level

Practioner Level

- QAI –Quality Assurance Institute

  CSTE - Certified Software Tester

  CMST - Certified Manager of Software Testing

- IIST – International Institute for Software Testing

  CSTP - Certified Software Test Professional

  CTM – Certified Test Manager

CAPÍTULO 5 – A IMPORTÂNCIA DO TESTE DE SOFTWARE

5.1 Conceito da Importância

A importância da atividade de teste se torna clara quando se tem uma visão ampla dos benefícios que são obtidos com a mesma. Não somente na área de tecnologia percebe-se que constantemente que estamos testando produtos, como bebidas, comidas, carros, roupas, entre várias outras coisas que são fundamentais no dia-a-dia. Agora também pode-se imaginar como seriam todos produtos que são consumidos diariamente, sem que fossem testados, por exemplo, um veículo com um defeito de segurança pode ocasionar um grave acidente. Outro exemplo que pode ser citado são os prazos de garantia que as empresas fornecem ao consumidor caso um produto apresente defeito, logicamente esses prazos não são aprovados somente para campanhas de marketing, mas também porque os produtos foram devidamente aprovados em severos testes de qualidade que comprovam a sua confiabilidade, caso ocorram muitas falhas depois que os produtos forem consumidos, é porque ocorreram erros nos métodos de teste que fizeram com que as falhas passassem despercebidas.

A partir do momento que a tecnologia começou a crescer rapidamente e virar peça fundamental para pessoas e empresas, tornou-se crucial contar com software de qualidade, ou seja, um software que fornece resultado correto quando alimentado com dados válidos e que identifica corretamente dados inválidos.

A atividade de teste além de ter por finalidade agregar qualidade ao produto antecipando a descoberta de defeitos, ela deve reduzir também o custo do projeto. Esse custo não está associado somente ao custo do desenvolvedor ou da construção do sistema, mas também após a implantação do projeto, como por exemplo, uma falha com o sistema de informação de um supermercado, isso certamente gerará um custo adicional para a empresa que presta suporte à loja, conseqüentemente pode-se imaginar o custo que próprio supermercado irá obter com seu software em estado inativo. Isso indica que atividade de testes não traz benefícios apenas ao desenvolvedor do software, mas também aos usuários e à finalidade para qual será utilizado o sistema.

Devido a real importância do Teste de Software, deve-se entender que essa atividade não pode ser utilizada para a simples demonstração que o programa funciona,

como pensam muitos desenvolvedores que acabam realizando testes óbvios, mas deve identificar o máximo de falhas do sistema para que estas sejam devidamente corrigidas.

Pressman (1995, p. 787), cita que atividade de teste não serve para demonstrar funcionalidade do software:

> Os desenvolvedores de software são, por sua própria natureza, pessoas construtivas. A atividade de teste exige que o desenvolvedor descarte noções preconcebidas da "corretitude" do software que ele acabou de desenvolver e supere um conflito de interesses que ocorre quando erros são descobertos.

A atividade de teste vem crescendo bastante de acordo com a evolução da tecnologia, porém entende-se pouco sobre a importância do teste de um sistema, assim ela passa despercebida por muitos gestores de software e conseqüentemente pelas empresas, que não a vêem como item cada vez mais estratégico para a condução de projetos de software de qualquer porte.

Os testes não proporcionam apenas softwares mais confiáveis, mas garantem que os investimentos em Tecnologia da Informação podem ser comprovados, dentro dos prazos, dos custos e qualidade determinada, reduzindo os riscos dos projetos.

5.2 A Evolução da Importância de Teste

Nos últimos anos a busca pela qualidade de software vem se destacando bastante no mundo da informática, e com isso vem crescendo também o investimento nos métodos de teste de software, tanto na área acadêmica como nas empresas.

Apesar do investimento nesta fase da construção de um software estar se desenvolvendo nos últimos anos, o surgimento dessa idéia não é tão recente assim. Na fase inicial da computação (décadas de 50 e 60), a qualidade era uma responsabilidade exclusiva do programador, sendo que padrões de garantia de qualidade para o software foram introduzidos no desenvolvimento de software sob contrato militar durante a década de 1970. A partir daí espalharam-se rapidamente para o desenvolvimento comercial de software. Nesta mesma década os testes de software tiveram seu grande "boom" quando Glenford Myers publicou dois livros que originaram o Teste de Software, foram eles "Software Reliability Principles and Practices" (Princípios da Confiabilidade de Software e Práticas) e "The art of software testing" (A Arte de Teste de Software), o primeiro foi considerado uma das bíblias da qualidade de

software, e o segundo foi que o originou o conceito desta fase do ciclo de vida de um sistema. Myers diz que o custo de correção dos defeitos tende a subir quando encontrados muito tarde, ou seja, quando um defeito é encontrado na fase de Manutenção, existe um custo muito maior para recuperar a funcionalidade do sistema do que quando o mesmo é encontrado logo na Análise de Requisitos ou no início do Projeto.

A fase de Teste, por não ter sido tratada com a importância devida, por muito tempo foi realizada pelos próprios programadores, e ainda sem métodos pré-estabelecidos, ficando a cargo de cada desenvolvedor testar sua aplicação sem uma visão geral do contexto do sistema, praticando os chamados “testes viciados” é difícil encontrar os erros que o usuário final pode encontrar posteriormente. Depois de percebida esta falha, as empresas começaram a repassar a atividade de teste ao usuário final, porém os mesmos não possuíam a técnica e o conhecimento necessário do produtos para executar a função, e aí surgem outros problemas decorrentes de testes mal elaborados. Depois dessas experiências, as maiores empresas desenvolvedoras de software perceberam que a fase de testes não era apenas um processo em que erros eram encontrados aleatoriamente, como por exemplo, provar um tempero de uma comida para comprovar se o sabor está bom, mas sim uma fase importante do ciclo de vida do sistema, que possui métodos e técnicas que exigem atenção especial para qualidade do produto final.

Com a percepção da importância da fase Teste de Software, foi implantada a equipe de testes, o meio termo entre o desenvolvedor e usuário final, onde se criou uma metodologia de testes que caminha em conjunto com o desenvolvimento, avaliando, validando e verificando todo o sistema.

5.3 Funcionalidade e Confiabilidade.

Um dos principais objetivos do Teste de Software é assegurar a funcionalidade do sistema, ou seja, estabelecer a confiança de que o software está sendo adequado a sua função, este deve estar correto para onde pretende-se utilizá-lo. O sistema deve estar de acordo com o previsto, já que deve atingir determinados objetivos.

Sommerville (2007, p. 342) coloca como principal o objetivo do teste, a confiabilidade do software:

> O principal objetivo do processo de verificação e validação é estabelecer confiança de que o sistema de software está adequado ao seu propósito, isso significa que o sistema deve ser bom o suficiente para o uso pretendido. O nível de confiabilidade exigido depende do propósito do sistema, das expectativas dos usuários do sistema e do atual ambiente de mercado.

Assegurando a funcionalidade do sistema, entende-se também que este é confiável, porém o nível de confiabilidade pode ser menor ou maior em alguns modelos de software, em alguns deles se exige mais precisão, total ausência de falhas, entre outros aspectos que envolvem validação e verificação do que em outros sistemas. Pode-se citar como exemplo um software utilizado na área da saúde, como de um médico cardiologista, este deve ser totalmente isento de erros para que o diagnóstico do paciente não tenha falhas, e retorne o resultado correto. Não é necessária precisa identificação de falhas em um sistema de chat interno de uma empresa, onde as falhas menos drásticas que possam existir não influenciam demasiadamente no trabalho do usuário, e estas podem ser corrigidas de acordo com que são encontradas. Esta diferença de tratamento não indica que alguns sistemas devem ser testados e outros não, isso apenas demonstra que existem sistemas que devem ser extremamente precisos, e por isso, utilizam mais da metodologia de testes do que outros.

Nos últimos anos os usuários de softwares em todo mundo vêm exigindo mais funcionalidade e confiabilidade dos softwares que eles utilizam, onde destaca-se para as empresas a importância da fase de teste. Anteriormente muitos usuários aceitavam passivamente as falhas de um software, pois entendiam que seus os benefícios apresentados pelos mesmos são maiores do que suas desvantagens, o que pode ser entendido, já que vários desses usuários provavelmente faziam o mesmo trabalho de forma muito mais complexa sem a tecnologia. Acostumados com a informática os usuários estão mudando esse conceito, e com a concorrência ainda mais acirrada entre desenvolvedores de software, admite-se pouco essas falhas, por isso as empresas desenvolvedoras de sistemas tendem a investir muito em validação, verificação e teste de seus produtos.

Com pouco investimento de uma empresa desenvolvedora de software em seus produtos nesta área de testes, obtêm-se logicamente produtos mais baratos, porém de pouca confiabilidade, assim estes devem ser adquiridos pelo restante de usuários “conformados” com falhas em sistemas de informação, porém esta fatia de consumidores no mercado tende a diminuir.

Pressman (1995, p. 768), relata a importância da confiabilidade de um programa de computador:

> Não há dúvida de que a confiabilidade de um programa de computador é um elemento importante de sua qualidade global. Se um programa deixar de funcionar repetida e freqüentemente, pouco importa se os outros fatores da qualidade de software são aceitáveis.

O termo Confiabilidade de Software cresce a medida que os sistemas vêm ficando mais complexos com o passar dos anos, essa complexidade se deve à inúmeras funções que um sistema pode ter, por isso, é essencial que ele seja funcional e confiável pois diversas operações dependem do mesmo para serem realizadas. A confiabilidade de software anda de mãos dadas com a verificação de software. As técnicas de testes estão associadas às características de confiabilidade necessárias.

Para dar mais importância a área de Teste de Software deve-se entender que o sistema não deve falhar no momento em que se mais precisa dele, e isso realmente pode ocorrer, e danificar todo um processo que depende muito ou até mesmo exclusivamente daquele software, já que ele foi implementado para obter mais agilidade, precisão, redução de custos, melhores resultados, integração, entre vários outros aspectos que podem ser prejudicados quando o software deixa de atender o resultado esperado.

5.4 Detecção de Defeitos

Como se sabe, a atividade de testes é realizada para se evitar defeitos, estes defeitos podem originar sérios problemas para o meio em que o software está atuando, esses problemas podem ser perda de informações, defeitos em processos importantes de uma empresa, retrabalho para as pessoas, prejuízos financeiros, dificuldade de comunicação e integração, entre vários outros transtornos que um sistema falho causa.

É possível até mesmo classificar os defeitos que podem ocorrer em um sistema, pois identificá-los é importante para outras fases do projeto, além de projetos novos:

- **Defeitos de interface com o usuário**: são aqueles apresentados diretamente ao usuário do sistema, ou seja, podem ser visualizados e identificados na prática por qual pessoa que opera o sistema, existe a necessidade de ser técnico, desenvolvedor para identificar esses erros para que sejam corrigidos. Esses

erros prejudicam a funcionalidade, ao não realizar no sistema o que usuário espera, a usabilidade, quando se obtêm uma dificuldade de navegação no software, dificultando sua praticidade, pode-se ter também um defeito de desempenho, quando sistema é lento para a finalidade desejada, ou um defeito de saída quando resultados apresentados são diferentes dos necessários para que atinjam a especificação do sistema.

- **Defeitos no manuseio de defeitos***:* esses defeitos são erros que o sistema encontra caso nele exista "brechas" para que os defeitos ocorram, por isso, no sistema deve tratar os dados que estão entrando, como uma forma de "prevenção dos defeitos", para que estas informações não sejam processadas inadequadamente. O programa deve identificar também os erros que são gerados e coletados em uma base de dados para que sejam corrigidos posteriormente, o que é chamado de "detecção de defeitos". Os sistema também deve ter as instruções de como tratar um defeito que é encontrado, o que é chamando de "reparação de defeitos".
- **Outros tipos de defeitos***:* esses são defeitos que podem ocorrer que não estão especificados acima, como defeitos de limites, onde o sistema não trata corretamente valores extremos ou fora de limites, defeitos de cálculos, defeitos de inicialização e fechamento, defeitos de controle de fluxo, defeitos de carga, onde o programa pode não suportar um pico de serviço em um determinado momento (estresse), ou até mesmo defeitos de controle de versão.

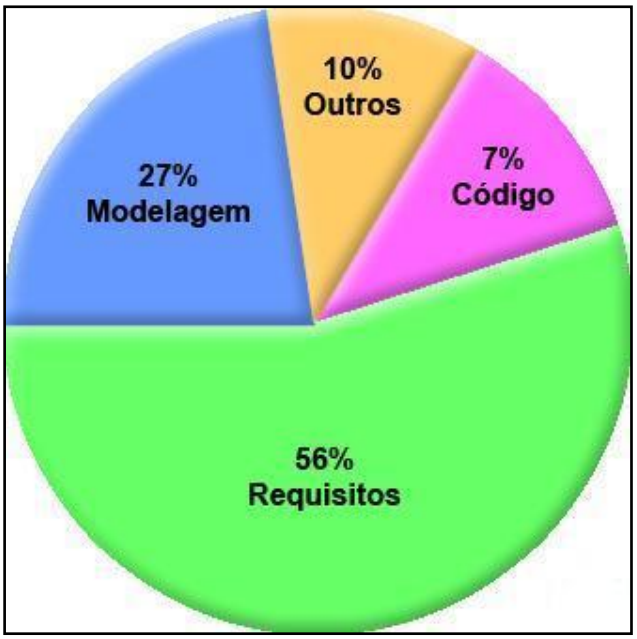
Neste tema, também pode ocorrer o termo faltas, segundo James F. Peters (2001, p. 500) os termos "faltas" e "falhas" têm distinção:

> "Dizemos que um sistema de software possui uma falta se para algum dado de entrada o comportamento do sistema está incorreto, ou seja, é diferente daquele incluído na especificação do software. Para cada execução do sistema de software onde seu dado de saída está incorreto temos uma falha de software. Portanto a falha é uma conseqüência de determinada falta que foi deixada no software, conforme aparece nos diagramas de fluxo"

Destaca-se desta forma a importância das atividades de testes em cada fase do sistema para solução de defeitos, já que estes defeitos podem vir de várias fases do

projeto. Aplicando esses métodos, é possível visualizar onde estão se concentrando mais as falhas do software, assim estudos comprovam que mais da metade das falhas estão na fase de Análise de Requisitos, onde os dados do cenário são coletados e organizados para uma visão teórica do software. O Gráfico 2 demonstra o percentual de falhas encontradas nas fases de um projeto, conforme esses estudos:

Gráfico 02 – Percentual de Falhas por Fase



Fonte: Paulo Tozelli

Deve-se destacar com esses estudos que, os testes de software não são apenas aplicados para que erros sejam encontrados, devolvidos a equipe de desenvolvimento e depois testados novamente, mas, além disso, a equipe de teste deve apontar de onde esses erros são provenientes, para que seja feito um estudo melhor elaborado e grande parte dessas falhas não voltem a ocorrer em projetos futuros.

Levando em consideração esse tipo de importância para as atividades de teste, fica claro que uma falha será encontrada somente uma vez em determinado projeto, analisada a sua origem, identificado o motivo de sua aparição, corrigida neste projeto e aplicada essa correção aos projetos seguintes que sejam semelhantes. Essa aplicação parece óbvia vista de forma simples, mas tratando-se em uma dimensão maior (muitas pessoas e várias equipes), estudos mal realizados fazem com que falhas já conhecidas se repitam insistentemente, aumentando drasticamente o custo de um projeto e seu tempo

de realização. Este processo não indica que em projetos futuros semelhantes não precisarão ser realizados esses testes, e que assim o trabalho da equipe de testes será menos importante, mas ocorrerá uma grande diminuição de retrabalho no desenvolvimento, no custo de um projeto, no tempo de construção do software, assim a equipe de testes irá trabalhar para encontrar situações ainda não conhecidas e melhorando gradativamente a qualidade dos softwares.

Pressman (1995, p. 786), indica o impacto de custos associados a falhas de software:

> O destaque crescente do software como elemento de sistema e os ‘custos’ envolvidos associados às falhas de software são forças propulsoras para uma atividade de teste cuidadosa e bem planejada. Não é incomum que uma organização de software gaste 40% do esforço de projeto total em teste.

Estudos como esse também indicam onde deverão ser investidos mais recursos no desenvolvimento do software, ou seja, caso a fase de Análise de Requisitos é aquela que mais proporciona falhas ao software, indica-se que podem ser investidos mais recursos nesta fase para que estes erros sejam prevenidos. Investir em uma área falha não acarretará mais custos para empresa, já que é mais custoso corrigir falhas de projeto de um projeto em andamento ou terminado, o que é chamado de retrabalho, do que trabalhar melhor no desenvolvimento do software, e diminuir assim o percentual de falhas de desenvolvimento.

5.5 Redução de Custos

A realização de testes possui também seus custos para ser aplicada, esses custos podem variar de acordo com a forma que sua aplicação é feita, por isso deve-se levar em consideração a importância da atividade de teste nesse aspecto, se existirá mais custo investindo em tecnologia de teste de software ou se a manutenção de defeitos do produto tornará mais caro que esse investimento.

Custo em um desenvolvimento de um projeto pode ser dividido em três partes distintas:

- Prevenção: é tipo de custo associado à prevenção de falhas. O investimento é realizado potencialmente na construção do projeto, tecnologia, pessoal

capacitado, aplicação de métodos e padrões, organização e planejamento são fundamentais para que o projeto fique pronto de forma correta logo na primeira vez. Este é um investimento alto, porém bastante sustentável para longo prazo.

- Avaliação: Trata-se do teste do produto, a sua revisão, inclui gastos com a inspeção, e equipe preparada para a sua realização. Desta forma o produto é avaliado antes de ser entregado ao usuário.
- Falha: Custo decorrente de uso de um produto com falhas, o mais alto dos custos está aqui, pois torna o retrabalho ainda mais difícil, além dos prejuízos que já foram obtidos com seus erros.

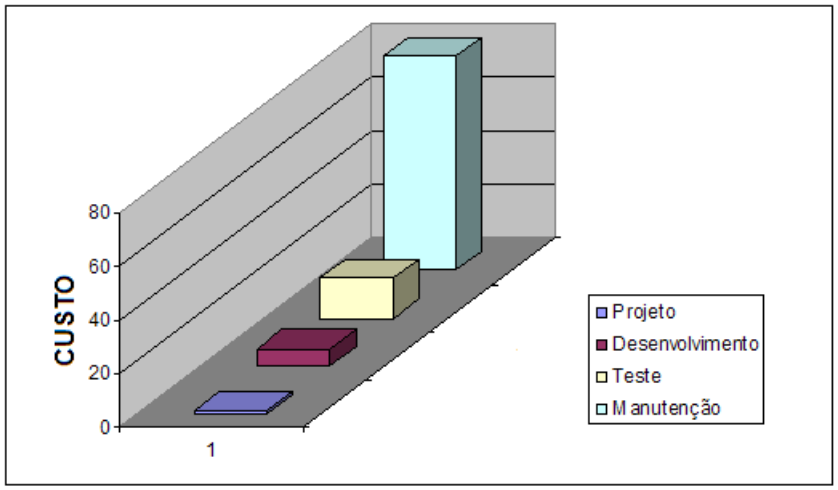
Esses três tipos definem onde estão as falhas mais custosas ao projeto, ou seja, quanto mais tarde um erro for localizado, mais caro será para corrigi-lo, esse custo pode ser até mil vezes maior.

Roger S. Pressman (1995, p. 737), demonstra um exemplo de custos obtidos ao detectar erros em um projeto:

> "Para ilustrar o impacto de custo da detecção de erros feita precocemente, consideramos uma série de custos relativos que se baseia em dados de custos reais compilados de grandes projetos de software [IBM81]. Suponhamos que um erro descoberto durante a fase de projeto custe 1,0 unidade monetária para ser corrigido. Em relação a esse custo, o mesmo erro, descoberto logo antes que as atividades de teste se iniciem, custará 6,5 unidades; durante os testes 15 unidades; e, após o lançamento entre 60 e 100 unidades."

Levando em consideração a citação de Pressman, segue o Gráfico 3 que representa o mesmo exemplo:

Gráfico 03 – Custo por fase de Desenvolvimento



Fonte: Pressman

Como demonstrado anteriormente, estudos indicam que a maior parte das falhas de um sistema são provenientes da Análise de Requisitos, assim imagine-se que se todas as falhas de um projeto forem encontradas após a entrega do produto, o custo de correção das mesmas será muito grande, já que a maior parte dos erros estão nos requisitos, onde deve-se obter o retrabalho de todo o projeto. Se esses erros são identificados rapidamente, o custo é muito menor.

A Figura 14 demonstra investimentos realizados em testes, o retorno e benefícios obtidos com isso:

Figura 14 – Tabelas de Investimentos

Estrutura de Teste	Teste Informal	Teste Manual	Teste Automatizado
Pessoal	0	60000	60000
Infra-estrutura	0	10000	10000
Ferramentas	0		12500
Total do Investimento	0	70000	82500
Defeitos Encontrados Pela Equipe de Desenvolvimento			
Defeitos Encontrados	250	250	250
Custo dos Defeitos	2500	2500	2500
Defeitos Encontrados Pela Equipe de Teste			
Defeitos Encontrados	0	350	500
Custo dos Defeitos	0	35000	50000
Defeitos Encontrados em Produção			
Defeitos Encontrados	750	400	250
Custo dos Defeitos	750000	400000	250000
Custo da Qualidade			
Conformidade(investimento em melhorias)	0	70000	82500
Não Conformidade(custo total dos defeitos encontrados)	752500	437500	302500
Total do Custo da Qualidade	752500	507500	385000
Retorno do Investimento	**Não há**	**350%**	**445%**

Fonte: Rex Black Article

Conforme a Figura 14 demonstrada, identifica-se que o retorno do investimento será maior à medida que ocorre o investimento em testes, ou seja, quanto mais for melhorado o processo de teste, serão melhores os resultados financeiros. As vezes, desenvolvedores de sistemas não se preocupam tanto com a atividade de teste, devido alto investimento que deve ser realizado no projeto, porém esses mesmos desenvolvedores não estão avaliando os custos com os defeitos que irão surgir posteriormente e assim o que parecia ser mais rápido, menos trabalho e custo, se torna mais demorado, com mais retrabalho e maiores custos.

CAPÍTULO 6 – A EQUIPE DE TESTE

6.1 Função da equipe

Como visualizado anteriormente na história do Teste de Software, muitas vezes a aplicação de testes era realizada pelos próprios desenvolvedores (programadores) do sistema, ou então essa função era delegada aos próprios usuários que iriam utilizar o software, ambos não possuíam técnicas ou conhecimentos necessários para que as falhas fossem encontradas. Foi diante dessa situação que foi criada uma equipe específica para a aplicação do Teste de Software.

No desenvolvimento de projetos, não somente de software, inconscientemente existe um conflito de interesses dentro das equipes, que faz com que o próprio desenvolvedor não perceba as inconsistências que aquele projeto pode obter, como por exemplo, um engenheiro civil, que se orgulha de uma construção sua terminada, mas não consegue enxergar nela melhorias que poderiam ser feitas. A mesma situação ocorre a construção de um sistema, o programador do sistema identifica todos os atributos positivos do projeto, porém, psicologicamente, não possui a capacidade de revelar na mesma apresentação possíveis erros que provavelmente estão ali.

A equipe de testes pode ser implantada de várias formas, de acordo com a necessidade dos testes, normalmente esta equipe é mais utilizada na fase final de testes, já que a própria equipe de desenvolvimento (analistas e programadores) é responsável pelos os testes por módulos, assegurando que as funções criadas estão funcionando de acordo com a normalidade, logo depois, na maioria das vezes, esta mesma equipe testa o sistema integrado, aí sim a equipe especializada na fase de Teste “entra em cena” para identificar possíveis inconsistências que não foram localizadas nas fases anteriores dos testes.

Pressman (1995), define o papel da equipe de testes da seguinte forma:

> O papel de um grupo independente de teste (Independent Test Group – ITG) é remover os problemas inerentes associados ao fato de deixar que o construtor teste o que foi construído. Os testes independentes suprimem o conflito de interesses que, de outra forma, poderia estar presente. Afinal de contas, o pessoal da equipe que compõe o grupo independente é pago para descobrir erros.

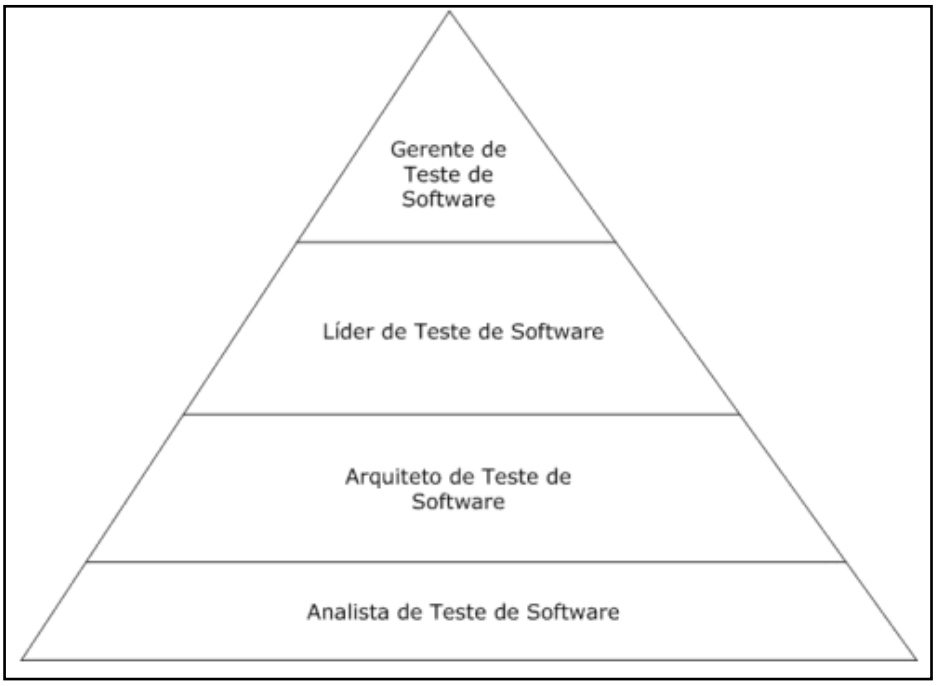
Durante o processo de testes, a equipe de desenvolvimento não deve se afastar do trabalho que a equipe de testes exerce, pelo contrário, os programadores devem acompanhar o processo, apontar os pontos mais importantes e estarem sempre a disposição para corrigir o mais rápido possível as inconsistências encontradas, para que novos testes sejam realizados.

Os profissionais da área de teste, muitas vezes, inconscientemente, podem não ser bem vistos pelos profissionais do desenvolvimento, por localizarem erros em seus projetos, mas são essenciais na garantia da qualidade do produto que está sendo testado. Profissionais desta área qualificados tecnicamente e em negócios geram menor custo para o desenvolvimento do software, como será visto posteriormente.

6.2 Formação da equipe

Uma equipe de testes pode ser vista na prática de várias formas, mas não muito diferente da Figura 15, onde demonstrada a hierarquia de uma equipe de testes definida em empresas especializadas em qualidade de software atualmente:

Figura 15 – Hierarquia da Equipe de Teste



Fonte: TestExpert

O Gerente de Teste de Software tem como função primordial a iniciação do projeto de teste a ser realizado no produto a ser testado. Suas qualificações e competências se assemelham a um gerente de projetos típico: elaborar o plano do projeto de teste, aquisição de novos recursos, orçamento, riscos, prazos, elaboração de relatórios, limitações do escopo do projeto de teste e outras atividades gerenciais como constante comunicação com sua equipe, controle e monitoração das atividades, geração de métricas para alimentar indicadores, etc.

O Líder de Teste é o profissional responsável pela condução dos testes e pela equipe de testes. Geralmente é um profissional com alto grau de conhecimento em ciclos de vida de testes, automação de testes, ambientes de testes. O Líder de Testes ajuda o gerente de Testes a elaborar três relatórios básicos para o acompanhamento do Projeto: Relatório de Status, Relatório de Progresso e Relatório de Desempenho.

O Arquiteto de Teste é o profissional responsável por montar e garantir o correto funcionamento da infra-estrutura e ambientes de teste. É também a pessoa que cuida da instalação das ferramentas e que garante o correto funcionamento das mesmas.

O Analista de Teste é responsável por traduzir os requisitos do sistema em modelagens, cenários e casos de teste. Os defeitos encontrados por este profissional são encaminhados à equipe de desenvolvimento e o mesmo deve garantir que os mesmos foram corrigidos através do Teste de Confirmação.

Para formação de uma equipe de testes, os responsáveis devem obter os seguintes requisitos:

Requisitos Técnicos:

- Conhecimentos gerais sobre as práticas e princípios de engenharia de software;
- Conhecimento das práticas e princípios de Teste de Software;
- Conhecimentos básicos sobre métodos e estratégias de teste;
- Conhecimentos em Planejamento, Design e Execução de testes;
- Conhecimentos de redes de computadores, banco de dados, sistemas operacionais;

- Conhecimentos de gestão de configuração;
- Conhecimentos dos vários tipos de documentação em teste de software;
- Habilidades para definir, coletar e analisar medidas de teste de software;
- Conhecimentos básicos em ferramentas de automação;
- Noções sobre qualidade de software.

Requisitos Gerenciais:

- Organização e planejamento;
- Motivação;
- Saber trabalhar com clientes e usuários;
- Saber trabalhar em equipe e resolver problemas de conflitos entre os membros da equipe;
- Habilidades para uma boa comunicação.

No Brasil, já é realidade que existam nas empresas de grande porte grandes equipes de testes de até 200 funcionários, e por isso, essa seleção de pessoas para formação pode ser bastante complexa, para que a equipe seja eficaz.

O primeiro passo dessa seleção das pessoas é obter o suporte da alta gerência para desenvolver as atividades de testes, assim como em qualquer outro projeto dentro de uma empresa, é necessário que a gerência reconheça a importância da atividade de testes para a mesma, caso contrário torna-se inviável a realização do processo, já que exige-se investimentos razoáveis, ferramentas de trabalho, local, maquinários. Esse apoio deve ser obtido também para que todos reconheçam a importância da equipe de testes para o projeto, e que a mesma trará benefícios a longo prazo, e não a curto prazo como muitos pensam.

Depois desse apoio, é importante formalizar e institucionalizar a atividade de teste na empresa, definir a existência desse grupo de colaboradores que são essenciais para a qualidade de software. A equipe de teste passa a fazer parte do plano de carreira da empresa.

Em terceiro, vem a necessidade de se definir as qualidades que requer cada cargo que deve ser ocupado dentro da equipe, ou seja, instituir os requisitos mínimos dos cargos para inscrição dos candidatos as vagas.

Elaborar a descrição de cargos é importante para que seja definido exatamente o que colaborar irá realizar, esse processo é dividido em algumas partes. Descrição do título, onde é definido o título da vaga (Gerente de Testes, Analista de Testes, etc.). Definição do salário de acordo com o mercado de trabalho, as qualificações, leis trabalhistas, entre outros. O local onde será realizada a função para determinado cargo. E as qualificações que o candidato deve ter.

Depois de todas as definições, são escolhidos os melhores candidato e entrevistados, geralmente são realizadas ao menos duas entrevistas, sendo a primeira com o departamento de Recursos Humanos, e posteriormente com o próprio gerente da área de testes ou de desenvolvimento, onde podem ser aplicados testes para avaliar o conhecimento técnico do candidato.

E por fim, são selecionados os candidatos escolhidos para cada cargo para trabalharem na empresa, é aí que se obtêm a Equipe de Teste de Software.

CONCLUSÃO

O principal objetivo desse trabalho foi apresentar a importância da atividade de teste de software, não apenas no contexto de desenvolvimento, mas também nos benefícios que uma empresa desenvolvedora de software alcança em seus resultados utilizando a atividade de teste. Na análise geral apresentamos todos os processos para o desenvolvimento de software, em ênfase a fase de teste, os tipos de testes, e os modelos de teste. De todas as atividades, envolvidas no desenvolvimento de software, o investimento na atividade de teste é a que traz maior retorno, tanto financeiro como de imagem.

Observamos que o foco das empresas, é a procura por qualidade de seu produto, devido às exigências do mercado, a competitividade e difusão da sua imagem. Porém estudos mostram que a maioria das empresas envolvidas nesse mercado não enfatiza a atividade de teste devido sua complexidade, custo de implantação e dificuldade em formação de equipes.

Dados comprovam que o não investimento na atividade de teste, pode trazer complicações financeiras, e um fator importante, a falta de qualidade. Um sistema além de um produto é uma ferramenta de trabalho indispensável nas organizações.

Entre os trabalhos futuros é necessário um estudo para visualize o custo de uma equipe de teste e comparar ao retorno alcançado no investimento em uma equipe de teste. Esse estudo pode trazer dados importantes para cálculos de precificação de software e índices de retrabalho.

REFERÊNCIAS

FERNANDES.Agnaldo Aragon.Gerencia de Software Através de Métricas 1 ed. São Paulo: Atlas,1995. 421 p.

MICROSOFT. Virtualização**.** disponível em <http://www.microsoft.com/brasil/servidores/virtualizacao/promise.mspx> . Acesso em 10 outubro 2008

PETERS, F.James. Engenharia de Software. 3 ed.São Paulo: Elsevier, 2001. 602 p.

PFLEEPGR, Shari Lawrence. Engenharia de Software. 2 ed. São Paulo: Pearson, 2004, 537 p.

PRESSMAN, Roger. Engenharia de Software. 3. ed. São Paulo: Makron, 1995. 1056 p.

ROCHA, Ana Regina Cavalcanti. MALDONADO, José Carlos. WEBER, Kival Chaves. Qualidade de Software. 1 ed. São Paulo:Pearson, 2001. 303 p.

SANTARÉM, José Eduardo. Virtualização: O impacto na Tecnologia da Informação. In Semana de Tecnologia da Informação, 1, 2008, Marília.

SOMMERVILLE, Lan.Engenharia de Software. 6 ed. São Paulo: Pearson, 2003. 592 p

SOMMERVILLE, Lan.Engenharia de Software. 8 ed. São Paulo: Pearson, 2007. 552 p

TESTEXPERT. Formando Equipes Eficientes de Teste de Software. Disponível em http://www.testexpert.com.br/?g=node/101. Acesso em 3 de outubro de 2008.

TOZELLI, Paulo. Análise do Teste de Software. disponível em http://imasters.uol.com.br/artigo/9817/des_de_software/analise_do_teste_de_software_-_parte_02/>. Acesso em 9 de dezembro de 2008.

ZERO-DEFECT. Vagas. Disponível em <http://www.zero-defect.com.br/pt/oportunidades/vagas/zero-defect-vagas>. Acesso em 10 de novembro 2008.

APÊNDICE

Para avaliação do mercado de Teste de Software atualmente no Brasil, foi realizada uma pesquisa entre 43 profissionais da área de TI, com a seguinte questão e as seguintes alternativas:

Questionário

Na empresa onde você trabalha, existe uma equipe exclusiva de Teste de Software?

- Uma ótima equipe, muito valorizada pela empresa
- Uma boa equipe, porém mal valorizada
- Equipe de teste composta por iniciantes (ou estagiários)
- Os testes são realizados pelos próprios desenvolvedores
- Não existe equipe de teste

FUKUMORI, Anderson Tadayoshi; SANTOS, Ismaías Gracino; MORRO, Tiago May.

A Importância da Atividade de Teste no Desenvolvimento de Software/ Anderson Tadayoshi Fukumori, Ismaías Gracino dos Santos, Tiago May Morro; orientador(a): Adalberto Sanches Munaro. Marília, SP: [s.n.], 2008.

66 f.

Trabalho de Curso (Graduação em Administração de Empresas) – Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, Marília 2008.

1. Requisitos 2. Desenvolvimento 3. Teste 4. Software 5. Qualidade. 6. Confiabilidade.

CDD: 005.14