Centro Universitário de Brasília
Instituto CEUB de Pesquisa e Desenvolvimento - ICPD

BÁRBARA FLORÊNCIO DA SILVA

AVALIAÇÃO DE PROCESSOS DE TESTE DE SOFTWARE PARA UMA EMPRESA PRIVADA

Brasília
2015

BÁRBARA FLORÊNCIO DA SILVA

AVALIAÇÃO DE PROCESSOS DE TESTE DE SOFTWARE PARA UMA EMPRESA PRIVADA

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para obtenção de Certificado de Conclusão de Curso de Pós-graduação *Lato Sensu* em Governança em Tecnologia da Informação.

Orientador: Prof. Paulo Rogério Foina

Brasília
2015

BÁRBARA FLORÊNCIO DA SILVA

AVALIAÇÃO DE PROCESSOS DE TESTE DE SOFTWARE PARA UMA EMPRESA PRIVADA

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Pós-graduação *Lato Sensu* em *Governança em Tecnologia da Informação*.

Orientador: Prof. Paulo Rogério Foina

Brasília, 01 de junho de 2015.

Banca Examinadora

_____

Prof. Dr. Paulo Rogério Foina

_____

Prof. Dr. Fabiano Mariath D`Oliveira

Dedico este trabalho aos meus pais que sempre me proporcionaram apoio incondicional.

AGRADECIMENTO(S)

Agradeço à Deus, por me dar uma vida saudável e sempre com muitas bênçãos;
Aos meus pais, pela educação que me foi dada e pela dedicação de toda a vida;
Aos amigos Marivone, Victor, João Ricardo, Tatiane e Everson pela cumplicidade, parceria e amizade;
Ao meu noivo Cássio que esteve ao meu lado em todos dias de estudo me dando força e ajudando sempre que possível;
Ao meu orientador Paulo Foina pela paciência, disponibilidade e ajuda constante;
Ao querido colega de trabalho Denys que me incentivou e me ajudou a conseguir uma boa oportunidade de participar deste curso.

A ciência de hoje é a tecnologia de amanhã (Edward Teller)

RESUMO

A realização de testes no desenvolvimento de softwares é fundamental para a obtenção da qualidade, visto que os clientes se tornaram cada vez mais exigentes devido à complexidade de seus interesses na área tecnológica. Os objetivos deste estudo envolvem classificar os processos de testes de software em função da sua usabilidade em empresas de diversos portes e; mapear as dificuldades em adotá-los. Para a metodologia deste estudo foi realizada inicialmente uma pesquisa bibliográfica com revisão teórica e características operacionais sobre os processos de teste de software mais usados por empresas de pequeno, médio e grande porte. Também foram feitas entrevistas com gestores de desenvolvimento sobre as dificuldades para a adoção plena dos processos de testes de software nessas empresas. A pesquisa indica que apenas a empresa de grande porte possui área específica para a realização de testes, bem como profissionais certificados nesta área. Em relação às dificuldades em implantar os processos existentes, a pequena empresa considera que seu maior obstáculo é obter maior aceitação das interferências externas, enquanto a empresa de médio porte necessita de evolução na programação orientada a testes; e a empresa de grande porte relatou que enfrentou dificuldades por conta do modelo de documentação que utilizava nos projetos.

Palavras-chave: Teste. Software. Empresas. Dificuldades.

ABSTRACT

The testing in software development is critical to the achievement of quality, as customers become increasingly demanding due to the complexity of their interests in technology. The objectives of this study involve classify software testing processes according to their usability in companies of different sizes and; map the difficulties in adopting them. For the methodology of this study was initially conducted a literature review with theoretical and operational characteristics of the software testing process commonly used by small, medium and large companies. Interviews were also done with development managers about the difficulties for the full adoption of software testing processes in these companies. Research indicates that only a large company has a specific area for testing and certified professionals in this area. Regarding the difficulties in implementing the existing processes, the small business believes that his biggest hurdle is to get greater acceptance of outside interference, while the medium-sized business needs evolution in programming oriented tests; and the large company reported that faced difficulties because of the documentation model that used in the projects.

Keywords: Test. Software. Companies. Difficulties.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

INTRODUÇÃO

Atualmente, a busca por qualidade está presente em quase todos os tipos de produtos e serviços que são consumidos por clientes de diferentes setores. No setor de software e serviços de software isso não é diferente. Clientes procuram saber da possível qualidade dos produtos e serviços que a organização de software poderá entregar. A visibilidade da qualidade de um software é “muito dificilmente percebida e para isso as organizações de software se adequam a modelos de qualidade com referência e bastante reconhecidos para que o mercado possa enxergar seus produtos e serviços como de alta qualidade” (ARANTES, 2007, p. 07).

Produtos de software, como qualquer outro tipo de produto, demandam um processo para sua confecção. Para Arantes (2007, p. 7) “quanto mais alta é a qualidade desse processo, maior é a probabilidade de o produto final ser um produto de qualidade”. Para isso foram desenvolvidos vários modelos e normas que podem ser utilizados para definir processos de software de qualidade, dentre eles: ISO 9000:2000 (ISO, 2000), ISO 12207 (ISO/IEC, 1998), ISO 15504 (ISO/IEC, 2003), CMMI (SEI, 2001) e MPS.BR (SOFTEX,2006).

Os processos de software, conforme modelos de referência, podem ser divididos em fases, e para se desenvolver um software com qualidade exige-se grande dedicação e comprometimento dos envolvidos no projeto em todas as fases, pois uma etapa depende da outra, e caso haja falha em alguma, as demais podem ficar comprometidas. Melhorias só podem ser propostas quando os envolvidos mantêm um nível de responsabilidade elevado no cumprimento de metodologias sistemáticas que alcancem uma padronização exata das técnicas utilizadas para o desenvolvimento dos softwares, pois isto assegura a utilização de medições confiáveis, indispensáveis para a avaliação do produto final, permitindo inclusive a sugestão de possíveis melhorias a cada ciclo (PFLEEGER, 2004).

Diversas vezes, o cronograma de um projeto é definido de acordo com as necessidades do cliente, não levando em consideração o prazo real necessário para se desenvolver determinado software. Conforme demonstrado no relatório Chaos do The Standish Group International (2013) grandes projetos possuem 10 vezes mais

chances de fracassar por completo do que projetos menores, exigindo que a organização perceba seu valor logo no início do ciclo de vida dos grandes projetos para identificar as recompensas e benefícios. Para tanto, o relatório ainda revela que não se pode confundir o estabelecimento de cronogramas, etapas e atividades específicas com a determinação de projetos menores.

Por estes motivos, a fase de testes, que é uma das fases de um projeto de desenvolvimento de software, fica comprometida, geralmente, por ser uma das últimas etapas e, caso se tenha gasto um prazo maior nas etapas anteriores, a de testes fica ou com um prazo bem menor ao que foi definido inicialmente ou por algumas vezes, sem prazo, comprometendo assim a entrega do projeto na data inicialmente definida pelo cronograma aprovado pelo cliente. Em grande parte dos projetos não existe um profissional dedicado exclusivamente para realizar os testes de software; quem geralmente absorve essa função é ou o analista de negócios ou o próprio desenvolvedor, o que pode acarretar em procedimentos viciados e distorcidos na visão da solução desenvolvida, não fazendo a validação de todo o contexto do software.

O presente estudo se propõe a compreender como se dá o processo de verificação da qualidade do software nas empresas privadas através dos diversos processos testes sob a perspectiva das práticas do CMMI.

Os objetivos do presente trabalho são:

a) Classificar os processos de testes de software em função da sua usabilidade em empresas de diversos portes e;

b) Mapear as dificuldades em adotá-los.

Para alcançar esses objetivos foi feito um levantamento bibliográfico com revisão teórica e características operacionais sobre os processos de teste de software mais usados pelas empresas. Também foram feitas entrevistas com gestores de desenvolvimento, de três empresas de diferentes portes, sobre as dificuldades para a adoção plena dos processos de testes de software nessas empresas. As empresas foram escolhidas em função da disponibilidade de dados para análise e pelas respectivas relevâncias no mercado regional.

O presente trabalho está estruturado em 3 capítulos.

No primeiro capítulo, apresentamos uma introdução sobre engenharia de software, o modelo CMMI, os processos de testes de software e como as empresas lidam com os processos de testes. No segundo capítulo é descrito como foram feitas as entrevistas com os gestores, e as dificuldades em adotá-los. Por fim, no terceiro capítulo são apresentadas as conclusões do trabalho.

1 TESTES DE SOFTWARE

Cabe ressaltar que a área da Tecnologia da Informação e Comunicação (TIC) sempre manteve uma missão de extrema importância no segmento empresarial, principalmente devido ao fato de que as organizações vêm se deparando com a atual concorrência altamente acirrada, onde a maioria das mesmas está utilizando ferramentas e processos tecnológicos para reduzir custos, minimizar riscos, elevar seu potencial de produtividade e aumentar a qualidade de produtos e serviços.

Historicamente:

> No início do desenvolvimento de software, a atividade de teste era encarada como a simples tarefa de percorrer o código fonte e corrigir problemas já conhecidos. Essas correções eram feitas pelos próprios desenvolvedores, não existindo recursos específicos para essa atividade. Devido esse motivo, os testes eram realizados somente muito tarde, quando o produto já estava quase ou totalmente pronto. Apesar de essa situação estar associada a uma prática muito ruim de desenvolvimento de software, ela ainda continua presente em muitas organizações (SILVA, 2010).

O começo dos anos 80 ficou marcado pela expansão na indústria de computadores os primeiros princípios voltados para a qualidade de software, visto que desenvolvedores e testadores passaram a trabalhar juntos para melhorar as características do software. Apesar disso, foi somente nos anos 90 que as etapas de testes foram devidamente praticadas, já que muitos testes ficavam somente no papel pelo fato de serem inviáveis. A partir da década de 90, muitos puderam ser executados, proporcionando o aumento da produtividade e qualidade no desenvolvimento de softwares (SILVA, 2010).

Este crescimento tem contribuído com a construção de novos potenciais empreendedores, porém tem aumentado à complexidade dos sistemas implantados e ferramentas utilizadas, exigindo que as organizações se demonstrem mais flexíveis através do uso de modelos mais eficientes. A conformidade das características com os requisitos empresariais torna-se fundamental para o alcance da qualidade e consequentemente para a melhoria do processo de teste.

De acordo com Falbo (2005) o software possui um ciclo de vida que abrange etapas que devem ser executadas durante o projeto de desenvolvimento, necessitando da associação de técnicas ferramentas e critérios de qualidade, assegurando uma base sólida para o mesmo. Estas fases envolvem:

- planejamento: responsável pelo fornecimento de uma estrutura que possibilita ao gestor a identificação de estimativas razoáveis dos recursos utilizados, atividades praticadas, custos envolvidos e prazos a serem cumpridos.

- análise e especificação de requisitos: onde é realizado o levantamento de requisitos, permitindo o refinamento do escopo. Isto assegura melhor entendimento do software a ser desenvolvido pelo engenheiro de software e maior domínio do problema.

- projeto: esta fase é responsável por incorporar requisitos tecnológicos aos elementos essenciais do sistema que foram estabelecidos na etapa anterior, e necessitam ser aperfeiçoados. Com isso, torna-se preciso estabelecer a arquitetura geral do software de acordo com o modelo construído na fase de análise de requisitos.

- implementação: nesta etapa cada unidade do software é implementada e detalhada.

- testes: são realizados diversos tipos de testes para garantir a confiabilidade do sistema. Normalmente os principais envolvem teste de unidade, de integração e de sistema, sendo de fundamental importância manter os resultados documentados.

- entrega e implantação: após o teste do software, o mesmo é colocado em produção, devendo atender os requisitos básicos solicitados pelo usuário.

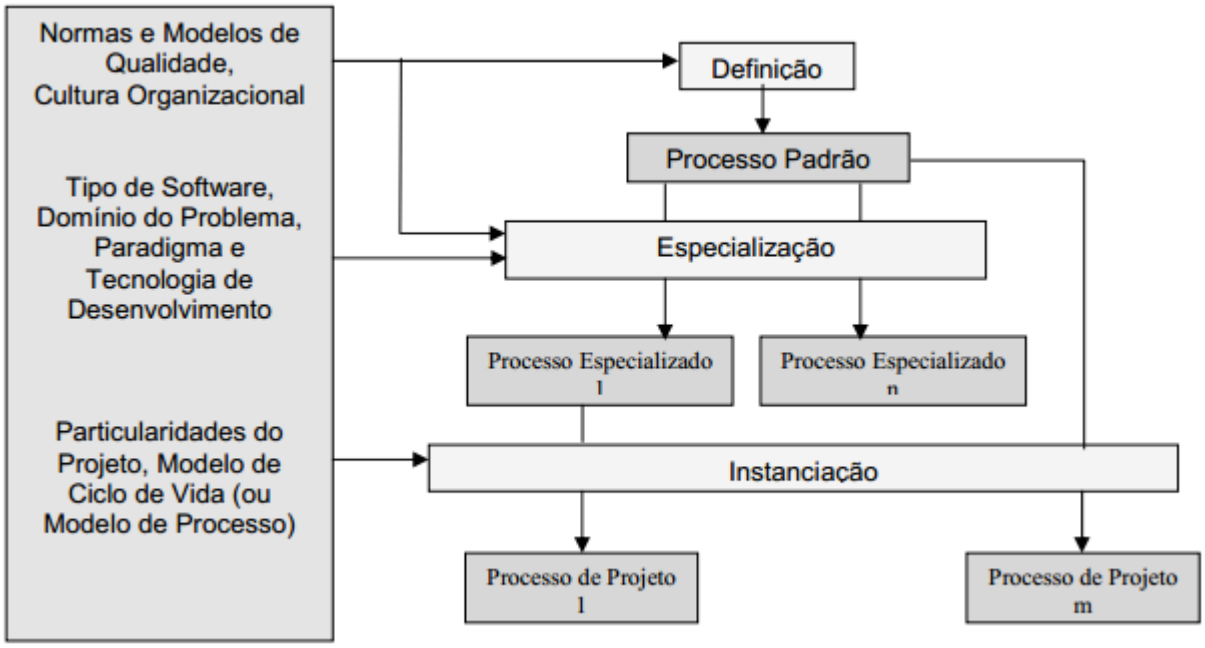
- operação: nesta etapa os usuários utilizam o software no ambiente de produção.

- manutenção: nesta etapa poderão ser feitas mudanças em decorrência da adaptação voltada para a funcionalidade adicional do usuário.

Com base nestas etapas, Falbo (2005) ainda afirma que é possível utilizar normas e modelos de qualidade em uma abordagem de definição de processos em

níveis, estabelecendo processos para projetos específicos, considerando as particularidades de cada projeto, conforme demonstrado na figura abaixo:

Figura 1 – Modelo para Definição de Processos em Níveis



Fonte: Falbo (2005, p. 09)

Falbo (2005) explicita que a distribuição de esforço por atividade deve ocorrer paralelamente ao processo de alocação de recursos, sendo possível construir uma rede de tarefas otimizada, já que consegue-se associar a cada uma delas o esforço necessário para que as mesmas possam ser efetivadas de modo satisfatório. A tabela a seguir demonstra a distribuição de esforço em cada atividade do processo de software, incluindo o de teste e entrega:

Tabela 1 - Distribuição de esforço em cada atividade do processo de software

Planejamento	Especificação e análise de requisitos	Projeto	Implementação	Testes e entrega
Até 3%	De 10 a 25%	De 20 a 25%	De 15 a 20%	De 30 a 40%

Fonte: Falbo (2005, p. 17)

Diante desta importância, Mazzola (2010, p. 82) afirma que:

> [...] o esforço despendido para realizar a etapa de teste pode chegar a 40% do esforço total empregado no desenvolvimento do software. No caso de programas que serão utilizados em sistemas críticos (aqueles sistemas dos quais dependem vidas humanas, como controle de vôo e a supervisão de reatores nucleares), a atividade de teste pode custar de 3 a 5 vezes o valor gasto nas demais atividades de desenvolvimento do software.

Para o autor, o mero cumprimento das especificações técnicas e utilização de metodologias e ferramentas impostas pela Engenharia de Software não assegura a qualidade do software desenvolvido, necessitando dos procedimentos de teste para revisar os requisitos desejados e validar a codificação. Com isso, o objetivo principal do teste de software é identificar o erro, indo contra “a falsa ideia de que uma atividade de teste bem sucedida é aquela em que nenhum erro foi encontrado” (MAZZOLA, 2010, p. 82).

Por sua vez, Carosia (2004, p. 28) ressalta que:

> Pode-se dizer que, um software de boa qualidade produz resultados úteis e confiáveis na oportunidade certa; é ameno ao uso, é mensurável, corrigível, modificável e evolutivo; opera em máquinas e ambientes reais; foi desenvolvido de forma econômica e no prazo estipulado; e opera com economia de recursos. Qualidade de software é um conceito muito mais amplo do que o de um software correto e bem documentado, requerendo para ser conseguida, metodologias e técnicas de desenvolvimento específicas.

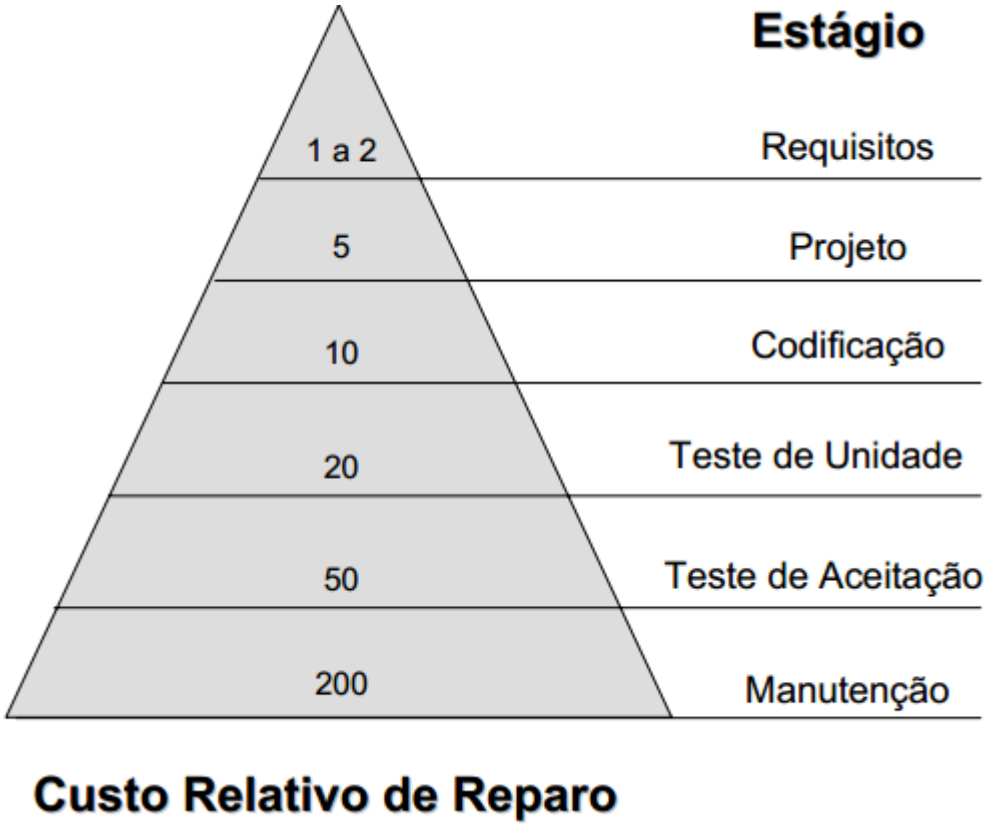
O autor ainda afirma que a qualidade também pode ser incluída ao produto final durante as etapas de testes, pois isto garantirá com que as características de qualidade sejam efetivadas. É preciso considerar que a qualidade deve permear as fases de desenvolvimento desde o início, devendo os profissionais estar cientes do nível de qualidade que desejam alcançar.

Em relação à importância de se praticar os testes, LEFFINGWELL (2005) ressalta que os mesmos são responsáveis por elevar consideravelmente a qualidade dos produtos desenvolvidos, prevenindo defeitos e assegurando as validações necessárias para atender os requisitos solicitados pelo usuário. Dessa forma, os testes garantem que o produto gerado atenda as expectativas do cliente,

funcionando corretamente no ambiente de produção sem comprometer a integridade do sistema.

O autor relata ainda que o custo de um problema pode chegar a ser 200 vezes maior se for reparado após a implantação, conforme demonstrado na figura abaixo:

Figura 2 – Custo de correção



Fonte: LEFFINGWELL (2005)

1.1 Engenharia de software

De acordo com Tigre (1984) os softwares estão ligados com uma série de programas associados com as operações de um sistema computacional, podendo ser categorizados em dois tipos básicos:

a) Software de sistema: são programas que controlam a execução de todos os programas encontrados no computador. Para Maziero (2008, p. 03) “o sistema operacional é uma estrutura de software ampla, muitas vezes complexa, que incorpora aspectos de baixo nível (como drivers de dispositivos e gerência de memória física) e de alto nível (como programas utilitários e a própria interface gráfica)”.

b) Software de aplicação: refere-se ao programa, ou o conjunto de programas, que realizam as atividades específicas pelo consumidor final, permitindo com que o hardware e o software permaneçam transparentes. Maziero (2008, p. 01) afirma que o software de aplicação é representado por programas destinados ao usuário do sistema, que constituem a razão final de seu uso, como editores de texto, navegadores Internet ou jogos.

Conforme afirma Pressman (2006), o software é o principal elemento para o bom funcionamento dos sistemas computacionais, pois controla esses sistemas e permitem que o mesmo trabalhe de maneira controlada otimizando as operações. O autor afirma que em muitos negócios os softwares são os verdadeiros diferenciais das empresas em relação a seus concorrentes. Diante disso, apresenta suas principais características:

- o software é criado e desenvolvido a partir de uma análise baseada na Engenharia de Software, e não de acordo com as manufaturas tradicionais;

- o software não possui prazo de validade como a maioria das mercadorias manufaturadas. O que ocorre é que o mesmo pode se tornar obsoleto, visto que a empresa necessitará de um software mais atualizado de acordo com suas necessidades;

- os softwares são desenvolvidos sob medida, para atender especificidades e características dos sistemas operacionais e dos clientes.

Segundo Pfleeger (2004) o processo de desenvolvimento de software muitas vezes ocorre sem a utilização de metodologias e etapas sistemáticas, que asseguram a integração das mesmas, aumentando a probabilidade de falhas, influenciando diretamente na qualidade do produto. Dessa maneira, para vencer obstáculos e combater problemas e falhas no desenvolvimento dos softwares, a Engenharia de Software busca apresentar princípios e diretrizes que auxiliem neste

processo, garantindo efetivamente a qualidade através da especialização dos profissionais envolvidos, utilização de técnicas formais, e compreensão das necessidades às mudanças do mercado atual.

Segundo Barbosa et al. (2000, p. 02):

> A Engenharia de Software pode ser definida como uma disciplina que aplica os princípios de engenharia com o objetivo de produzir software de alta qualidade a baixo custo. Através de um conjunto de etapas que envolvem o desenvolvimento e aplicação de métodos, técnicas e ferramentas, a Engenharia de Software oferece meios para que tais objetivos possam ser alcançados.

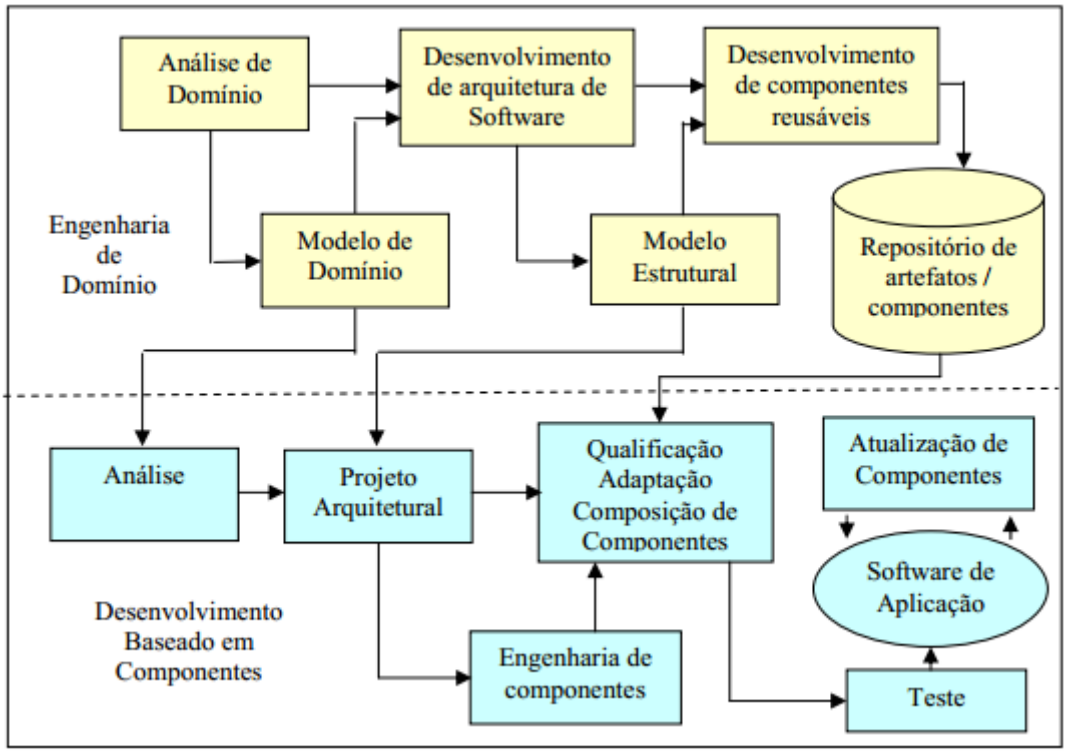
O principal objetivo da Engenharia de Software é proporcionar, à organização, o desenvolvimento de novos projetos, sendo possível atender a inúmeras solicitações de manutenção do cliente, devendo ser praticados em um prazo pré-estabelecido de maneira satisfatória a um preço competitivo que forneça uma margem lucrativa para ambas as partes. Com isso, é preciso enfatizar que o fator crítico disto se encontra no fato de que o desempenho da fábrica de software se apoia na dificuldade em gerenciar múltiplos projetos e processos, que podem ser interdependentes ou não, e que concorrem pelos mesmos recursos humanos disponibilizados pela fábrica, exigindo um esforço máximo de realocação (SILVA, 2005).

Para Fernandes e Teixeira (2004) as fábricas de software permanecem orientadas para os fundamentos da Engenharia de Software e Industrial, a fim de atender as diversas demandas produtivas, bem como os requisitos técnicos, tecnológicos e logísticos dos usuários e clientes do ponto de vista econômico e qualitativo. Além disso, utilizam a tecnologia de objetos, visando o desenvolvimento de uma biblioteca de componentes, assegurando a adoção de padrões de projeto (“design patterns”). Também faz uso de frameworks padronizados, objetivando estruturar os processos e controlar as operações de maneira contínua e repetitiva.

Pressman (2005) propôs um modelo de desenvolvimento de software que interfere no escopo da fábrica de componentes, impactando na Engenharia de Domínio, uma vez que a mesma ocorre paralelamente a Engenharia de Software da

organização. Com isso, é possível reutilizar os modelos de domínio, as estruturas e as bibliotecas dos componentes conforme apresentado na figura abaixo:

Figura 3 – Engenharia de Software baseada em componentes

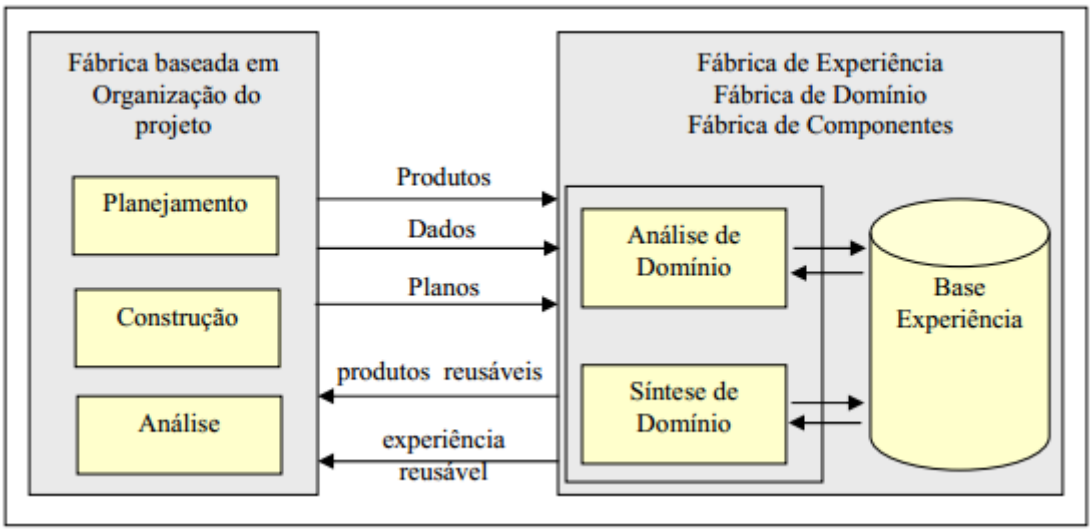


Fonte: Pressman (2005)

Com base na figura acima, Pressman (2005) destaca que a reutilização sistemática é fundamental para o desenvolvimento de novos processos da Engenharia de Domínio, como parte independente da Engenharia de Software. Cabe ressaltar que o produto principal é composto por elementos que podem ser reutilizáveis na construção de outros sistemas.

Conforme relatam Basili et al. (1992) os processos de produção de software envolvem uma vasta quantidade de relacionamentos e objetos que resultam em experiências complexas que necessitam ser incorporadas ao ambiente organizacional através da Engenharia de Software. Para tanto, estes autores propõem um modelo de framework orientado ao reuso, estruturado a partir de duas abordagens, onde a primeira permanece voltada ao projeto e a segunda denominada de fábrica de experiência, conforme apresentado na figura abaixo:

Figura 4 – Framework organizacional



Fonte: Basili et al. (1992)

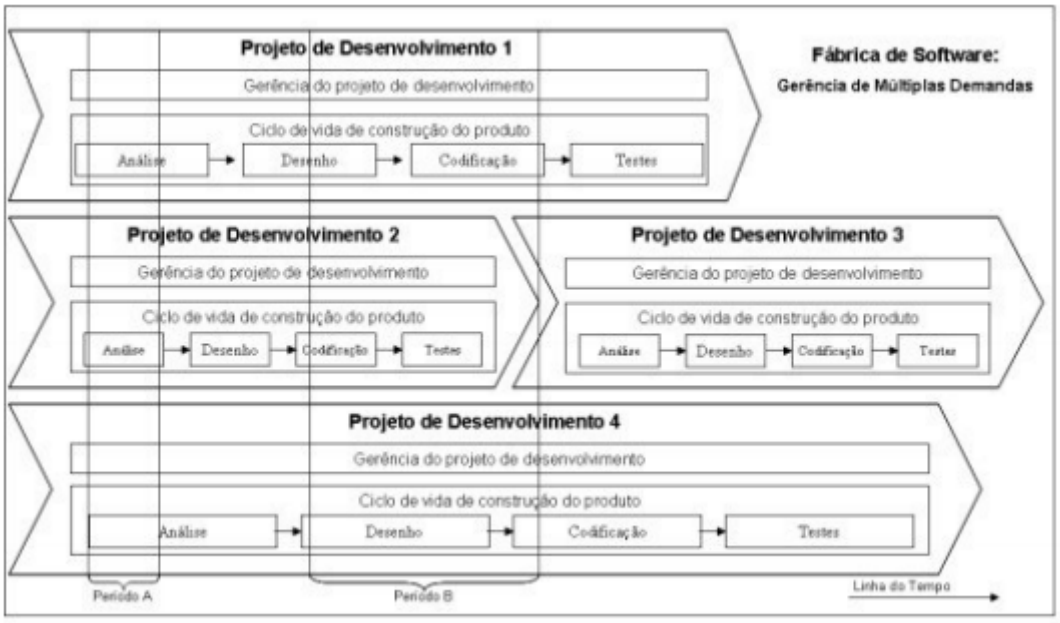
Através da figura acima, compreende-se que a fábrica de experiência busca apresentar uma abordagem responsável pela monitoração e análise dos projetos desenvolvidos, sendo possível elaborar ainda ações de reuso de diferentes tipos de experiências associados aos dados, processos, ferramentas e produtos, a fim de potencializar a reutilização dos componentes do software.

A arquitetura da fábrica de experiência pode ser apresentada a partir de dois níveis de abstração, a fábrica de domínio cuja finalidade é gerenciar as informações relacionadas ao processo de reutilização e a fábrica de componentes, cujo objetivo é produzir e manipular o conjunto de elementos que serão direcionados para a fábrica de projetos.

Do ponto de vista da Engenharia de Software, é fundamental a disponibilização de estruturas que permitam o gerenciamento de processos de múltiplas demandas, sendo possível estabelecer as prioridades e distribuir os recursos de modo mais efetivo e produtivo para a organização, atendendo com isso as questões de custos e prazos, reduzindo significativamente os riscos coletivos (ROCHA et al., 2001).

Silva (2005) propõe um framework de processos baseado no gerenciamento de múltiplas demandas e em modelos de melhores práticas:

Figura 5 – Fábrica de Software com múltiplas demandas



Fonte: Silva (2005)

Destinado à construção de novos produtos, Silva (2005) explicita que a fábrica de software é constituída pelos seguintes processos:

- Modelagem de negócios: onde é possível compreender a estrutura organizacional e sua dinâmica, e consequentemente entender a finalidade do software construído, identificando possíveis problemáticas e melhorias.

- Requisitos: estabelece uma relação benéfica entre cliente e organização através do estabelecimento e compreensão dos requisitos do sistema, fornecendo maior entendimento sobre os mesmos de modo que a mesma possa desenvolver estimativas de custo, prazo de entrega e definir uma interface.

- Análise e desenho: este processo possui o objetivo de transformar os requisitos em desenhos arquitetônicos do sistema, sendo possível atender o nível de desempenho desejado.

- Implementação: parte que organiza os códigos em camadas, implementando classes e objetos, e sobretudo, testando os componentes desenvolvidos como unidades, a fim de integrar os resultados.

- Teste: onde o software pode ser testado, os requisitos validados e possíveis melhorias podem ser sugeridas e praticadas.

- Garantia da qualidade: neste processo é possível “localizar e documentar defeitos na qualidade do software, verificar se os processos padrões da fábrica foram seguidos e se os produtos de trabalho foram gerados e armazenados corretamente” (SILVA, 2005, p. 03).

- Implantação: por fim, este processo providencia a instalação do software.

1.2 Modelo Capability Maturity Model Integration (CMMI)

O Capability Maturity Model Integration (CMMI) é baseado no desenvolvimento do Modelo de Maturidade da Capacidade ou Capability Maturity (CMM) e segundo Carosia (2004, p. 34) “é um modelo que baseia em um processo gradual, que leva as organizações a se aprimorarem continuamente, na busca de soluções próprias para os problemas existentes no desenvolvimento de software”. A autora ainda afirma que o responsável pelo desenvolvimento deste modelo foi o Software Engineering Institute (SEI) a fim de proporcionar um novo padrão de qualidade para as Forças Armadas no desenvolvimento dos softwares.

Historicamente:

> Em 1987, a partir do trabalho de Humphrey, o SEI lançou uma breve descrição de um ambiente de maturidade de processo de software e um questionário de maturidade para avaliar o estado corrente das práticas de software e identificar as áreas que necessitavam de melhoria. Em 1991, o SEI criou o CMM, como uma evolução deste ambiente. A partir do lançamento e divulgação da versão 1.1 do CMM, em 1993, o tema de melhoria do processo foi ganhando força (CAROSIA, 2004, p. 34).

As melhorias foram observadas devido aos estudos realizados pelas organizações com base no modelo CMM. Assim, as mesmas passaram a adotar uma estrutura mais eficaz que permitia o alcance de maior qualidade no processo de desenvolvimento de software, onde os benefícios foram realmente notórios.

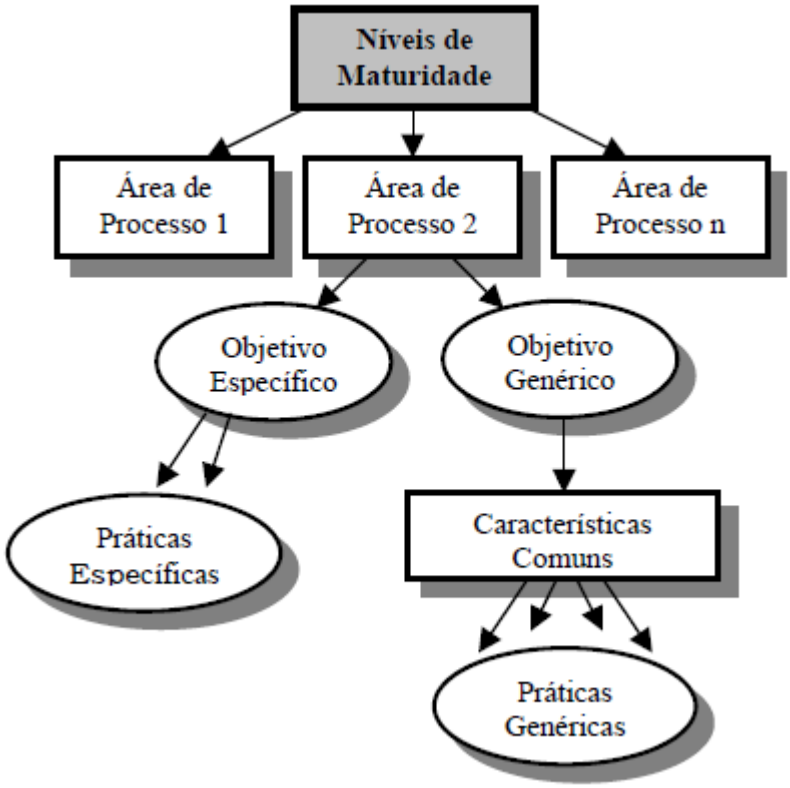
Em relação à sua maturidade, conforme relata Silva (2010), o CMM “baseia-se em um modelo evolutivo de maturidade, no qual as organizações partem

de uma total falta de controle e gerenciamento de processos". Este processo é denominado de maturidade organizacional, já que as organizações vão obtendo novas habilidades e potenciais de maneira gradativa, melhorando seus níveis de eficiência, alcançando maior maturidade. Isto auxilia as organizações a superarem as dificuldades e obstáculos na criação dos softwares.

Diante disso, o CMMI foi criado para melhorar as diretrizes e orientações do CMM. O modelo mais atual integra quatro disciplinas, dentre elas: Engenharia de Sistemas; Engenharia de Software; Desenvolvimento de Processo e Produto Integrado e Contratos de Fornecedores.

Para atender as disciplinas acima, os níveis de maturidade deste modelo constituem em conjuntos pré-definidos de processos, devendo todos os objetivos estarem integrados para o atendimento das áreas específicas. Os níveis de maturidade previstos pelo CMMI são: inicial, gerenciado, definido, quantitativamente gerenciado e otimizado.

Figura 6 – Níveis de maturidade do CMMI



Fonte: Carosia (2004, p. 48)

Entende-se que o objetivo do controle de qualidade proporcionado pela obtenção dos níveis de maturidade do CMMI nos processos de desenvolvimento de software é identificar etapas, técnicas e ferramentas que possam aumentar a confiabilidade do software desenvolvido. Além disso, com a identificação dos problemas nesta etapa é possível realizar as correções necessárias para garantir que os atributos empresariais sejam atendidos (OCHNER; MENDES, 2007).

Conforme relata Nomura (2008) no CMM, um dos principais objetivos dos processos de certificação é avaliar se as atividades desempenhadas se encontram documentadas, e se esta documentação é pertinente às mesmas. Com isso, o estabelecimento de processos de software torna-se imprescindível, a fim de alcançar os níveis de maturidade desejados. O autor relata ainda que a partir do nível 3, a importância da definição de processos é maior, necessitando da padronização dos processos de desenvolvimento de software. Enfatiza-se que diferentes projetos podem coexistir mantendo características específicas desde que permaneçam integrados aos objetivos organizacionais.

No Brasil, o processo de Melhoria de Processo do Software Brasileiro (MPS-BR) é vastamente utilizado pelas empresas, pois possui seus sete níveis de maturidade relacionados com os do CMM. Este modelo surgiu no intuito de definir e aperfeiçoar as ações de avaliação do processo de desenvolvimento de software, contribuindo com o alcance da qualidade do produto final. Essas diretrizes auxiliam micro, pequenas e médias empresas brasileiras no aprimoramento de seus negócios, permitindo que as mesmas possam comercializar os softwares seguramente no mercado nacional e internacional (MPS.BR, 2006).

> O MPS.BR tem como foco, atender às micro, pequenas e médias empresas de software brasileiras, com poucos recursos e que necessitam obter melhorias significativas nos seus processos de software. [...] tem como base os requisitos de processos definidos nos modelos de melhoria de processo e busca implantar os princípios de Engenharia de Software adequada ao contexto das empresas brasileiras, estando em consonância com as abordagens internacionais para definição, avaliação e melhoria de processos de software (MPS.BR, 2006, p.5).

Este modelo de referência possui requisitos que as organizações precisam atender para permanecer em conformidade com as exigências do mercado.

Tabela 2 – Níveis de maturidade do MPS-BR

Nível	Processos	Atributos de Processo	
A	Análise de Causas de Problemas e Resolução – ACP	AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP3.2, AP 4.1, AP 4.2 , AP 5.1 e AP 5.2	CMM Nível 5 - Otimizado
B	Gerência de Projetos – GPR (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2, AP 4.1 e AP 4.2	CMM Nível 4 - Gerenciado
C	Gerência de Riscos – GRI	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2	CMM Nível 3 - Definido
	Desenvolvimento para Reutilização – DRU		
	Análise de Decisão e Resolução – ADR		
	Gerência de Reutilização – GRU (evolução)		
D	Verificação – VER	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2	
	Validação – VAL		
	Projeto e Construção do Produto – PCP		
	Integração do Produto – ITP		
	Desenvolvimento de Requisitos – DRE		
E	Gerência de Projetos – GPR (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP3.2	
	Gerência de Reutilização – GRU		
	Gerência de Recursos Humanos – GRH		
	Definição do Processo Organizacional – DFP		
	Avaliação e Melhoria do Processo Organizacional – AMP		
F	Medição – MED	AP 1.1, AP 2.1 e AP 2.2	CMM Nível 2 - Repetível
	Garantia da Qualidade – GQA		
	Gerência de Configuração – GCO		
	Aquisição – AQU		
G	Gerência de Requisitos – GRE	AP 1.1 e AP 2.1	
	Gerência de Projetos – GPR		

Fonte: Barbosa (2008, p. 31)

Estes modelos são considerados fundamentais para os processos do ciclo de vida devido à necessidade de alcançar o mais alto nível dos mesmos em todas as etapas, obtendo um ótimo desempenho com as atividades e tarefas desenvolvidas no projeto.

1.3 Processos de teste de software

Inúmeros processos estão sendo propostos no intuito de contribuir com a projeção de casos de teste para um software, uma vez que esta atividade é bastante complexa e precisa ser praticada de modo eficiente, sendo possível demonstrar as funções operacionais e internas do produto desenvolvido, visando sempre o seu aperfeiçoamento (PFLEEGER, 2004).

Atualmente, existem inúmeras metodologias que promovem o teste do software. No entanto, as vastamente utilizadas e que proporcionam melhores resultados permanecem direcionadas para o uso de linguagens estruturadas, valorizadas principalmente para os sistemas orientados a objeto (ROCHA et al., 2001).

1.3.1 Estrutural e funcional

Segundo Pressman (2005) as técnicas de teste existentes englobam as diferentes perspectivas do software. Existem dois tipos de técnicas:

- a estrutural, denominada de teste caixa branca, cuja finalidade é avaliar o componente de software, sendo possível atuar diretamente sobre o código fonte, analisando as condições do sistema, o fluxo de dados, os ciclos e os caminhos lógicos efetuados. Os elementos avaliados nesta técnica podem variar de acordo com a tecnologia utilizada no processo de desenvolvimento do software.

Durante o processo de teste, o profissional terá acesso ao código fonte da aplicação, podendo desenvolver outros códigos possibilitando a mediação de bibliotecas e componentes, permitindo com isso a ampliação de análise para todas as possibilidades do componente de software.

- funcional, denominado de teste caixa preta, que consiste na avaliação comportamental do sistema a partir do fornecimento de dados de entrada,

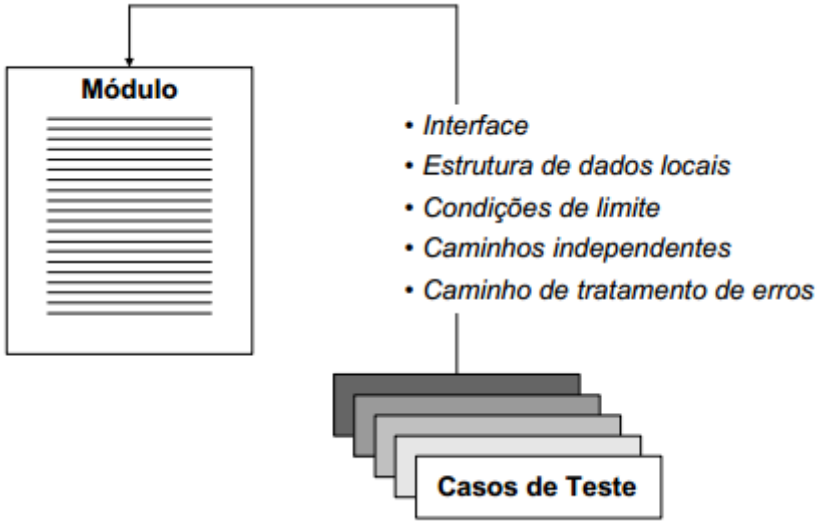
estabelecendo um comparativo com informações previamente conhecidas. Esta técnica pode ser aplicada em todos os níveis de teste, voltada para uma função interna, programa específica, componente ou funcionalidade (PRESSMAN, 2005).

1.3.2 Testes de unidade, integração, validação e sistema

Segundo Mazzola (2010) uma forma de subdividir o teste de software e facilitar a identificar de erros no programa é fazer uso dos testes de unidade, integração, validação e sistema. O teste de unidade possui o objetivo de verificar se há falhas nos módulos a partir das informações do documento do projeto detalhado do software, sendo considerado um teste de caixa branca, podendo ser aplicado paralelamente sobre outros módulos.

Em relação aos aspectos a serem testados, Mazzola (2010) afirma que inicialmente, a interface deste teste permite a busca dos erros através da passagem de informações para dentro e para fora do módulo, onde é composto por estruturas de dados locais que possuem a finalidade de garantir o armazenamento das informações, de modo a manter sua integridade durante a execução do módulo. As condições limite do teste verificam as tarefas executadas durante o processamento do módulo, enquanto os caminhos independentes são analisados visando maior segurança para que todas as instruções sejam aplicadas pelo menos uma vez. Por fim, os caminhos de tratamento de erros, caso sejam observados, efetivarão os testes, demonstrando o desempenho do módulo (figura 1).

Figura 7 – Teste de unidade



Fonte: Mazzola (2010)

Para Martins (2009) os testes de unidade buscam falhas em módulos, funções, classe, pequenos grupos de classes (clusters), componentes e serviços. Além disso, asseguram e mantem a finalidade de exercitar detalhadamente a unidade do sistema, sem perder a generalidade do mesmo, onde este teste poderá identificar a presença de interfaces incorretas de entrada e saída; problemas na integridade dos dados armazenados; condições de limite inadequadas prejudicando a operação do sistema, e falhas no tratamento de erros.

Dentre os testes de unidade que mais se destacam, os testes de interface é o primeiro a ser realizado, cujo objetivo é identificar qualquer anomalia referente às duvidas existentes nos demais testes, avaliando: a) a coerência entre o módulo e os parâmetros de entrada; b) os argumentos transmitidos aos módulos e os parâmetros de entrada; c) a consistência dos argumentos e a ordem de passagem das funções embutidas; d) a existência de referências a parâmetros não associados ao ponto de entrada atual; e) a mudança de argumentos; f) a consistência de variáveis globais dos módulos; e, f) as restrições dos argumentos (MAZZOLA, 2010).

O teste de estruturas de dados é utilizado para analisar todo o contexto da unidade de modo global, sendo possível avaliar o bom uso dos mecanismos

responsáveis por definir as estruturas de dados dos diferentes escopos. Cabe ressaltar que os erros mais frequentes deste teste envolvem digitação inconsistente, iniciação incorreta de variáveis; inconsistência nos tipos de dados; e underflow e overflow.

O teste de condições limite “busca verificar que o que foi definido a nível de projeto para as condições limite de uma unidade está sendo respeitado a nível de implementação” (MAZZOLA, 2010, p. 86). Por sua vez, o teste de caminhos de execução permite que sejam identificadas as seguintes falhas: a) precedência aritmética incorreta; b) operações envolvendo dados de tipos diferentes; c) inicialização incorreta; d) erros de precisão; d) representação incorreta de uma expressão; e) laços mal definidos; f) comparação de diferentes tipos de dados; e g) operadores lógicos utilizados incorretamente (MAZZOLA, 2010, p. 87)

O teste de integração identifica erros referentes ao processo de integração das diferentes unidades dos componentes do software. Martins (2009) afirma que o mesmo integra unidades já avaliadas, sendo capaz de descobrir problemas de interação e compatibilidade entre as mesmas.

Segundo Mazzola (2010, 84) “o fato de se ter analisado os módulos do software de forma exaustiva (através de procedimentos de teste de unidade), não há nenhuma garantia de que estes, uma vez colocados em conjunto para funcionar, não apresentarão anomalias de comportamento”. Com isso, este teste é o primeiro e mais importante a ser realizado sobre a unidade, já que avalia as falhas do processo voltadas para análise dos seguintes aspectos: a) a coerência entre argumentos do módulo e os parâmetros de entrada; b) a coerência entre argumentos passados aos módulos chamados e os parâmetros de entrada; c) a análise dos argumentos e da ordem estabelecida dos mesmos em suas funções desempenhadas; d) as modificações de argumentos de entrada e saída; e) a consistência de variáveis no decorrer dos módulos; e, f) a passagem de restrições sobre argumentos.

O quadro a seguir apresenta as falhas identificadas no processo de teste de integração:

Quadro 1 – Falhas de integração

• Falhas de interpretação: ocorrem quando a funcionalidade implementada por uma unidade difere do que é esperado. – B implementa incorretamente um serviço requerido por A. – B não implementa um serviço requerido por A. – B implementa um serviço não requerido por A e que interfere com seu funcionamento.
• Falhas devido a chamadas incorretas: – B é chamado por A quando não deveria (chamada extra). – B é chamado em momento da execução indevido (chamada incorreta). – B não é chamado por A quando deveria (chamada ausente).
• Falhas de interface: ocorrem quando o padrão de interação (protocolo) entre duas unidades é violado. – violação da integridade de arquivos e estruturas de dados globais. – tratamento de erros (exceções) incorreto. – problema de configuração / versões. – falta de recursos para atender a demanda das unidades. – objeto incorreto é associado a mensagem (polimorfismo).
• Problemas não funcionais: ocorre quando requisitos não funcionais são violados – Módulo B não tem o tempo de resposta esperado por A. – B lança exceções não esperadas por A.
• Incompatibilidades dinâmicas: ocorrem quando a ligação dinâmica entre as unidades do sistema é permitida. – Polimorfismo dinâmico entre classes. – Conexões dinâmicas entre serviços ou componentes.

Fonte: Martins (2009)

A abordagem mais utilizada para a prática dos testes de integração é a big bang:

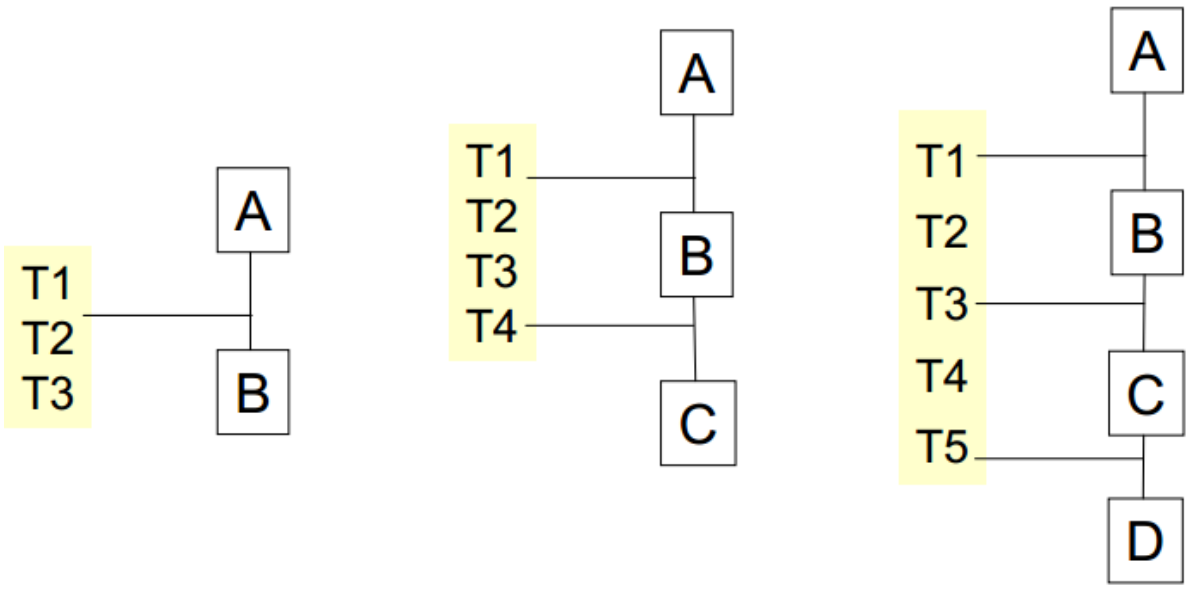
> [...] um processo de integração não-incremental na qual todos os módulos são associados e o programa é testado como um todo. Na maioria dos testes realizados segundo esta abordagem, o resultado não é outro senão catastrófico. A correção dos erros torna-se uma

> tarefa extremamente complexa, principalmente porque é bastante difícil isolar as causas dos erros dada a complexidade do programa completo. Mesmo quando alguns erros são detectados e corrigidos, outros aparecem e o processo caótico recomeça.

Através desta abordagem é possível desenvolver o programa de acordo com os requisitos desejados, testando o mesmo em etapas, sendo possível identificar os erros de maneira isolada, e consequentemente desenvolver soluções mais eficientes. Enfatiza-se que assim o teste de interfaces pode ser realizado ainda com maior eficácia.

Por sua vez, Martins (2009) afirma que pode ser utilizada ainda a abordagem incremental que faz uso de estratégias baseadas na estrutura do sistema com o uso do grafo de dependências e na arquitetura, podendo estar orientada por suas funcionalidades, conforme demonstrado na figura abaixo:

Figura 8 – Abordagem incremental



Fonte: Martins (2009)

Por sua vez, o teste de validação contribui com a estruturação do software, pois impacta na sistematização do mesmo em pacote. Com isso, é

possível verificar se o software é capaz de cumprir com suas funções adequadamente de acordo com suas especificações técnicas. Mazzola (2010) afirma que normalmente este teste pode apresentar dois resultados, dentre eles:

1 – as características e resultados obtidos atendem os requisitos desejados, se enquadram também nas especificações estabelecidas;

2 – as especificações não atendem aos requisitos e padrões, sendo listada uma série de falhas.

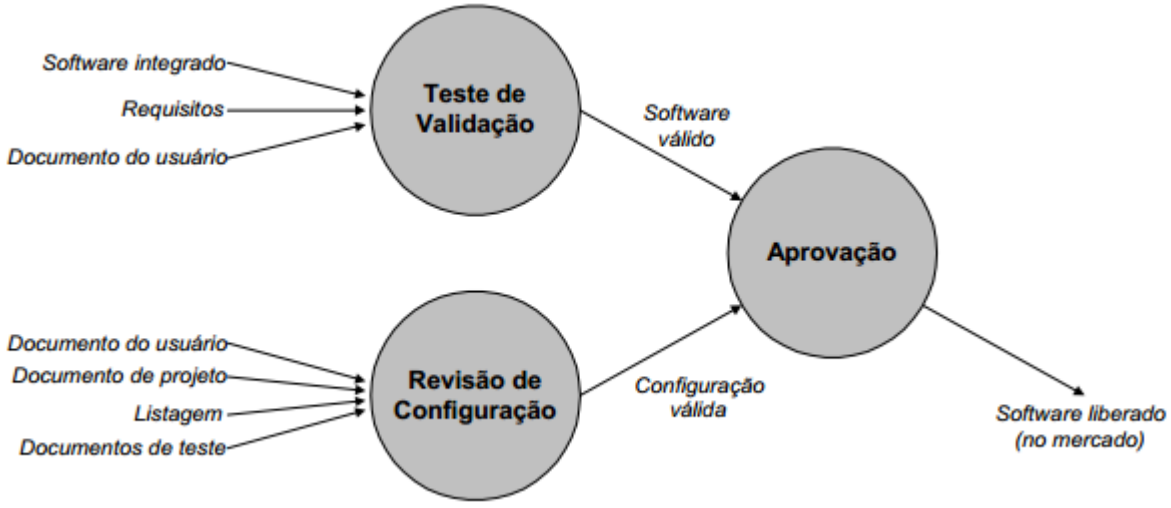
Com a observação das falhas e possíveis erros, torna-se fundamental que soluções sejam propostas para a reversão do quadro e eliminação das deficiências.

Cabe citar que:

> Um aspecto de importância no processo de validação é a chamada revisão de configuração. Ela consiste em garantir que todos os elementos de configuração do software tenham sido adequadamente desenvolvidos e catalogados corretamente com todos os detalhes necessários que deverão servir de apoio à fase de manutenção do software (MAZZOLA, 2010, p. 122).

Este processo pode ser facilmente visualizado a partir da figura a seguir:

Figura 9 – Revisão de configuração



Fonte: Mazzola (2010, p. 122)

É preciso citar que embora os profissionais de testes se esforcem para validar um software, é muito difícil prever as diversas maneiras que o mesmo poderá ser utilizado, principalmente em sistemas altamente interativos. Para tanto, testes de aceitação devem ser realizados no intuito de envolver o cliente e atender a demanda solicitada. Enfatiza-se que este processo pode variar entre poucas semanas há vários meses, de acordo com a complexidade do sistema ou natureza da aplicação e são permeados a partir dos testes alfa e beta.

Os testes alfa “são realizados com base no envolvimento de um cliente, ou numa amostra de clientes, sendo realizados nas instalações do desenvolvedor do software” (MAZZOLA, 2010, p. 123). Por sua vez, os testes beta “são conduzidos em uma ou mais instalações do cliente pelo usuário final do software” (MAZZOLA, 2010, p. 123).

Mazzola (2010) enfatiza a importância de se documentar os erros encontrados pelo próprio usuário, devendo os mesmos ser encaminhados ao desenvolvedor, a fim de encontrar as melhores soluções.

E por fim, os testes de sistema buscam verificar os atributos voltados para o funcionamento do software, permanecendo integrado ao hardware e outros elementos. Cabe ressaltar que conforme a complexidade do sistema testado, alguns requisitos e variáveis só poderão ser testados neste processo. Para tanto, várias modalidades de teste podem ser praticadas, podendo atender as necessidades organizacionais e a fim de testar as características computacionais do sistema (MAZZOLA, 2010).

Nesta categoria, o teste de recuperação é utilizado para avaliar o potencial do sistema em recuperar-se perante a ocorrência de erros, num tempo previamente determinado. Para que este processo seja possível, os erros são gerados artificialmente através de uma técnica denominada de injeção de falhas que possui a finalidade de analisar o potencial do sistema sob o ponto de vista da recuperação.

Outro teste utilizado é o de segurança, que:

> [...] visa garantir que o sistema não vai provocar danos recuperáveis ou não ao sistema pela sua própria ação. Por exemplo, num sistema de informação acadêmica, um aluno deve ter autorização de acesso para a visualização das notas obtidas nas disciplinas que cursou (histórico escolar), mas não deve ter a possibilidade de alterar seus valores (MAZZOLA, 2010, p. 124).

No teste de segurança, o profissional experimenta inúmeras ações, dentre elas:

- derrubar o sistema de modo geral;

- acessar informações confidenciais;

- alterar informações da base de dados;

- prejudicar o funcionamento do sistema;

- inserir vírus no sistema;

- sobrecarregar o funcionamento do sistema.

O teste de estresse verifica o comportamento do sistema em relação às situações de limite vivenciadas pelo mesmo sob diferentes concepções, sendo possível avaliar o limite da quantidade de usuários conectados ao servidor; quantidade de memória utilizada; versões de processadores necessárias; quantidades de bloqueios identificados (MAZZOLA, 2010).

Em relação ao teste de desempenho, Mazzola (2010, p. 125) afirma que:

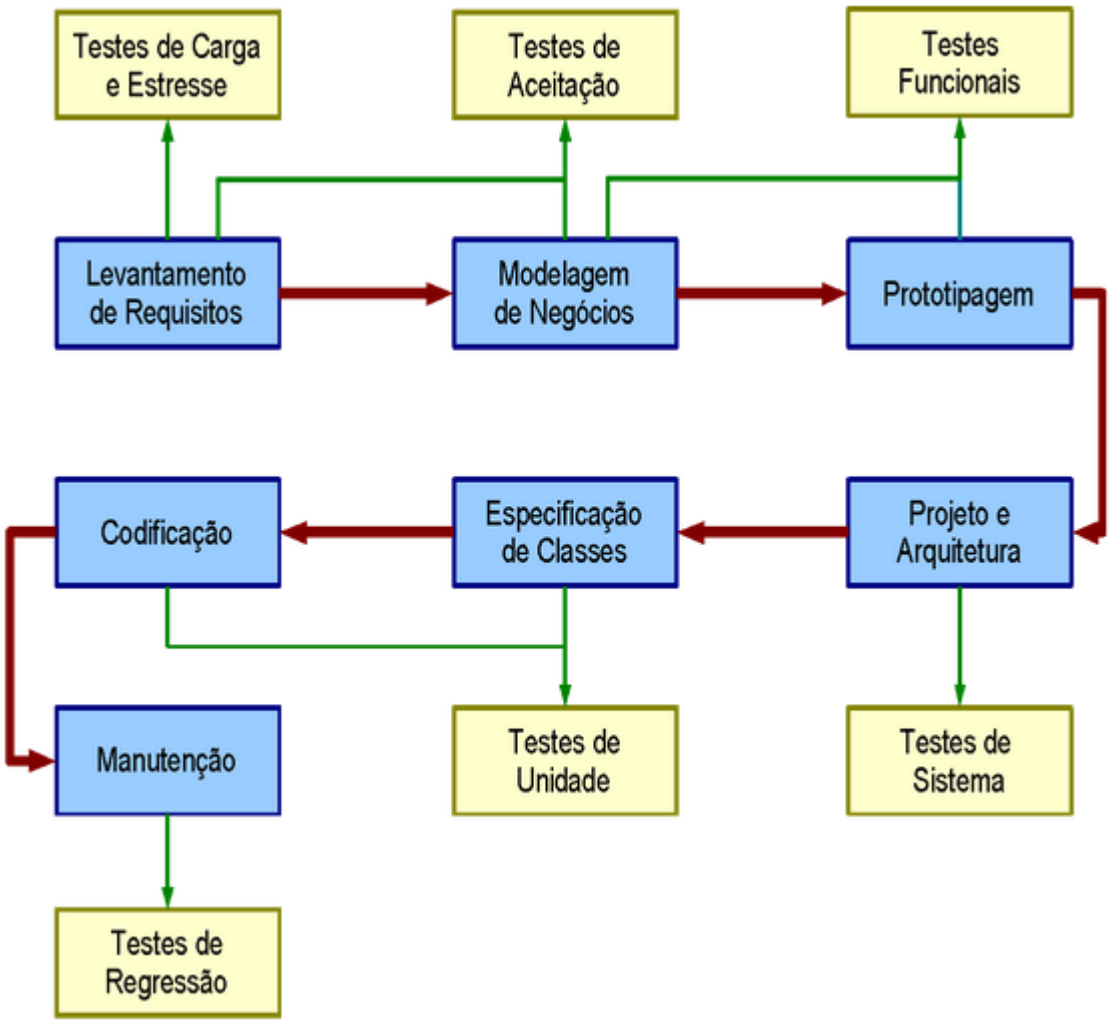
> [...] consiste em verificar se os requisitos de desempenho estão sendo atendidos para o sistema como um todo. Este aspecto, que pode não ter grande importância em alguns sistemas (quais?), torna-se crítico em aplicações envolvendo sistemas embarcados e sistemas multimídias, que pertencem à classe dos sistemas temporal. Nestes sistemas, o não atendimento a um requisito de tempo pode afetar de forma irreversível a função do sistema e, no caso de alguns sistemas (os chamados sistemas críticos), a não realização desta função ou o não atendimento a estes requisitos temporais pode resultar em prejuízos catastróficos (risco a vidas humanas ou grandes prejuízos financeiros).

Os testes de desempenho se apoiam na utilização de hardwares e softwares que contribuem com a medição dos recursos utilizados no sistema, a fim

de melhorar o processo de coleta de dados e de armazenamento das informações, assegurando o pleno funcionamento do sistema.

Com base nestes testes, Lozano (2014) propõe que seja seguido o esquema abaixo no desenvolvimento e realização de testes de software, podendo ser adaptados de acordo com as necessidades da empresa:

Figura 10 – Etapas de desenvolvimento e testes de software



Fonte: Lozano (2014)

2 ANÁLISE DE RESULTADOS

Os processos de testes de software descritos pelo CMMI se refletem atualmente no comportamento das empresas na busca em implantar ou mesmo melhorar o processo de teste utilizado. Ainda que as técnicas de teste de software mais utilizadas foram criadas por volta dos anos 80, as empresas têm uma grande dificuldade com a atividade de teste.

Para avaliar a qualidade do processo de software nas empresas privadas, como estão sendo aplicados e quais as maiores dificuldades na implantação dos processos, foram elaboradas algumas questões que ajudarão a identificar como as empresas estão em relação aos processos de testes.

Adotou-se como universo da presente pesquisa o formado por três empresas privadas, uma de pequeno porte, outra de médio porte e outra de grande porte. Assim como o SEBRAE (2014), consideraremos neste trabalho os seguintes critérios para se definir o tamanho da empresa:

- Pequena: de 10 a 49 empregados
- Média: de 50 a 99 empregados
- Grande: mais de 100 empregados

Para melhor avaliação, o questionário que foi aplicado nas empresas pode ser encontrado no apêndice deste estudo. As empresas foram selecionadas em função da relevância de cada uma no seu mercado de atuação e pela disposição em fornecer os dados para a pesquisa. Os critérios usados para selecioná-las foram: a) reputação no mercado local; b) quantidade de projetos realizados, e c) disponibilidade de dados. Selecionamos as empresas especializadas em desenvolvimento de software, em todo seu ciclo de vida, e que possuem equipes próprias de desenvolvimento.

2.1 Empresa de pequeno porte

O gestor desta empresa atua no ramo de desenvolvimento de softwares sob encomenda na organização JRSM Informática, que mantém 27 funcionários.

Esta organização não é composta por uma área dedicada à execução de processos de testes, pois os testes são realizados pela mesma equipe que implementa os produtos, portanto, nenhum dos profissionais possui certificação.

As etapas do processo de teste de software praticadas desde o planejamento envolvem o Teste de Funcionalidade cuja função é testar a funcionalidade geral do sistema em termos de regras de negócio (fluxo de trabalho), considerando tanto as condições válidas como as inválidas, e também o Teste de Segurança de Acesso voltado para a avaliação de todos os mecanismos de proteção embutidos no software, de fato, garantindo a proteção de acessos indevidos.

Os Testes efetuados pela empresa são: Testes unitários realizados pelo implementador; teste de sistema realizado pelo analista responsável pelo produto; e teste de aceitação realizado pelo cliente no ato da entrega do produto.

Em um mesmo projeto de desenvolvimento de software normalmente são executados 3 tipos de testes: os unitários, os de sistema e os de aceitação. Em relação aos documentos utilizados no planejamento e execução dos mesmos, a empresa afirma que faz uso do plano de teste, caso de testes, log de testes e termo de homologação.

Cita ainda que para ser aprovado, cada plano de testes define as condições mínimas que sofrem grande variação dependendo do cliente, da complexidade do produto que está sendo implementado, do tempo disponível para entrega do produto e dos recursos disponibilizados para implementação do projeto.

O tempo médio de planejamento para testes que compõem o cronograma original normalmente é de 16 horas, porém pode chegar a 40 horas para ser devidamente efetivado. Os softwares utilizados nas atividades de planejamento e realização de testes são o Team View e o Test Center da Microsoft.

Cabe ressaltar que na empresa de pequeno porte, os testes são realizados pelo analista do projeto e pelos desenvolvedores no caso de testes unitários. O percentual de rejeição pode chegar a 70% no primeiro ciclo de testes, 45% no segundo ciclo, e em seguida para 10%, chegando a zero nos ciclos seguintes.

Os testes são realizados conforme os ciclos de desenvolvimentos são concluídos e repassados para testes de aceitação quando não são identificados erros ou falhas pelo analista responsável pelo projeto.

Em virtude do atual tamanho e capacidade produtiva da empresa, o perfil exclusivo de analista de testes não é recomendado sendo necessário que este absorva as características de analista de sistemas.

Verificou-se na empresa de pequeno porte que o cliente somente é envolvido na entrega do produto ou em outros entregáveis.

Além disso, considerando o intuito de identificar as dificuldades em implantar os processos existentes, observou-se que o processo de testes ainda não está totalmente estabelecido nesta empresa e se encontra em processo de adequação. No entanto, durante o período de implantação, a mudança de cultura da equipe de desenvolvimento no que tange a aceitação das interferências externas resultantes no processo de testes tem se mostrado o maior obstáculo.

2.2 Empresa de médio porte

O gestor da organização de médio porte atua como gerente de TI na empresa WebAula S/A, que possui 62 funcionários.

Assim como a empresa de pequeno porte, a empresa de médio porte não possui uma área dedicada à execução e suporte aos processos de testes, portanto, também não disponibiliza profissionais certificados em testes de software. Dessa forma, conforme relatado, não planejam os testes, já que os mesmos são testados de acordo com as funcionalidades do mesmo e antes da entrega.

Dentre os testes realizados se destacam: de Configuração, de Integridade, de Segurança, Funcional, de Integração, de Performance (de Carga e de Estabilidade), de Usabilidade, de Caixa Branca, de Caixa Preta, de Regressão e de Manutenção. Cabe ressaltar que estes testes são realizados pelos próprios desenvolvedores, com exceção do teste de caixa preta. Todos estes testes são normalmente aplicados num mesmo projeto de desenvolvimento de software.

Como não planejam os testes, não fazem uso de qualquer tipo de documentação para facilitar a execução dos mesmos. No entanto, quando questionada sobre as condições mínimas para um teste de software ser considerado aprovado, a empresa considera que é preciso cumprir com os requisitos de aceitação descritos na história de usuário.

O gestor da empresa de médio porte também afirmou que como não realizam o planejamento, não é possível estabelecer um tempo médio para testes, pois eles fazem parte do Sprint, da definição de “software pronto”.

As ferramentas utilizadas nos testes envolvem: a SDK de desenvolvimento, o banco de dados, os próprios browsers, o fiddler, ferramentas próprias para acompanhamento dos servidores e a ferramenta on-line LeanSentry. O responsável final pelos testes é o coordenador de desenvolvimento, que em nível hierárquico se encontra apenas abaixo do Gerente do Setor.

Infelizmente, como a empresa não pratica o planejamento não é possível registrar incidências, portanto não possui a capacidade de avaliar o percentual de testes rejeitados ou analisar seu desempenho progressivo. O gestor afirmou também que não existe o interesse da empresa em contratar um profissional com perfil de analista de testes.

Verificou-se ainda que a empresa não realiza testes periodicamente e nem envolvem os clientes nos processos de testes, no entanto os mesmos sempre exigem que os produtos estejam funcionando perfeitamente.

Embora a empresa não cumpra com as ações de planejamento, seu gestor afirma que a mesma não mantém dificuldades para implantar os processos existentes, necessitando apenas de evolução na programação orientada a testes, TDD, o que ajudará muito na realização do teste de Caixa Branca.

2.3 Empresa de grande porte

A empresa de grande porte estudada é a Softtek tecnologia da informação, que possui em torno de 1.000 funcionários, onde o gestor entrevistado atua no ramo de consultor em TI.

Esta empresa possui uma área específica dedicada à execução e suporte aos processos de testes. Portanto, conta com profissionais competentes e qualificados que atuam neste processo, sendo 10 colaboradores, onde 3 deles possuem certificações em testes de software.

Quando questionado sobre quais as etapas abrangem o processo de testes de software praticadas pela empresa, o gestor afirmou que o primeiro passo envolve o mapeamento dos cenários com base na especificação de requisitos e caso de testes. Todo esse mapeamento é registrado e gerenciado pelo software TestLink. Após o mapeamento, os cenários levantados são executados pelo analista tester e caso o chamado seja aprovado, é encaminhado o relatório de execução por email aos responsáveis do chamado informando a devida aprovação. Caso seja encontrado um defeito, é encaminhado o mesmo relatório por email reprovando o chamado com as evidências dos defeitos em anexo junto com as métricas do defeito.

Dentre os testes efetuados neste processo se destacam: Teste Unitário, Teste Funcional, Teste de Setup e Teste de Regressão. Segundo o gestor da empresa, todos os tipos de testes mencionados no documento de Plano de Teste são executados durante o projeto de desenvolvimento de software.

Dessa forma, para apoiar o planejamento e execução dos testes, a empresa faz uso de documentos base como especificação de requisitos, caso de testes e as evidências de defeitos.

Quando questionado sobre as condições mínimas exigidas para que um teste de software possa ser considerado aprovado, o gestor relatou que após o mapeamento dos testes na ferramenta TestLink e a validação da massa de testes na base de dados referenciada no documento de Caso de Teste e considerando o

sucesso na realização dos testes, podemos considerar que a change request está aprovada.

Não é possível estabelecer um tempo médio para os testes devido a grande variabilidade de duração dos mesmos, porém na empresa de grande porte a estimativa normalmente fica acima da média de execução. Em relação ao esforço, o número é muito variável, visto que a empresa apresenta atividades de testes que partem de 4 horas até atividades que chegam às 200 horas, dependendo da complexidade do projeto.

Dentre os softwares usados para apoiar as atividades de planejamento e de realização de testes destaca-se a ferramenta TestLink para planejamento e gerenciamento dos cenários de testes.

Na empresa de grande porte, o responsável final pelos testes é o líder técnico da equipe.

Quando abordado sobre o percentual de resultados de testes rejeitados ou não aprovados desde o início deste ano, o gestor afirmou que inicialmente 98,15% das demandas que foram validadas pela equipe de testes foram rejeitadas.

Sobre as fases do desenvolvimento dos testes, o profissional relatou que inicialmente os testes são realizados pelos próprios desenvolvedores através de teste unitário, logo após o término do desenvolvimento. Após este passo, o chamado é encaminhado para a equipe da fábrica de testes, onde serão realizados todos os testes funcionais.

Como a empresa de grande porte possui um departamento específico voltado para a realização de testes, a contratação de um analista de testes é de bastante interesse. Dessa forma, o gestor ressalta que o profissional para atuar neste cargo deve manter uma boa percepção e ser proativo na operação do dia a dia, além de ter força de vontade em aprender, mantendo capacidade para enfrentar os desafios diários que a própria profissão lhe submeterá.

O cliente também se envolve nas fases que compõem o processo de testes, atuando no teste de aceitação (UAT) onde o ciclo de teste é contemplado no documento de log de teste.

A última pergunta buscou analisar quais foram às dificuldades em implantar os processos existentes e quais são as que ainda têm em implantar algum outro tipo de teste na empresa. Dessa forma, o gestor afirmou que inicialmente a empresa enfrentou dificuldades por conta do modelo de documentação que utilizava nos projetos. Para os testes, a organização considera muito importante que sejam utilizados processos e consultas que permitam gerar a massa de testes, pois as informações são mutáveis. Estas informações são a base necessária para um bom processo de teste.

CONCLUSÃO

O estudo permitiu compreender que o software é considerado um elemento fundamental para o bom funcionamento dos sistemas operacionais que são utilizados pelas organizações para melhorarem suas operações no ambiente organizacional, demonstrando importante papel no alcance de diferenciais no mercado, e consequentemente na obtenção de maior rentabilidade.

A empresa busca desenvolver programas e sistemas com qualidade, uma vez que necessita gerenciar múltiplos projetos e processos que exigem a realização de diferentes atividades, necessitando dos mais variados recursos humanos com qualificações específicas.

Com isso, o framework organizacional voltado para os processos de uma fábrica de software normalmente se encontra direcionado para o gerenciamento de múltiplas demandas, sendo possível alocar recursos e atividades de maneira eficiente, sendo possível atender os prazos, custos e objetivos pré-estabelecidos.

No entanto, o processo de desenvolvimento de softwares é bastante complexo, pois cada projeto deve estar voltado para o atendimento de necessidades específicas. Assim, demonstra-se que o desenvolvimento de testes é indispensável para o alcance da qualidade e atendimento dos requisitos solicitados pelo usuário, que devido à expansão das ferramentas tecnológicas e características do mercado torna-se cada vez mais exigente.

Diante disso, afirma-se que a qualidade do produto final se encontra relacionada com a qualidade de todos os processos realizados durante o desenvolvimento do software. Assim, existem inúmeros modelos de padrões internacionais e nacionais que oferecem normas e diretrizes para orientar as organizações no alcance da qualidade total.

O modelo CMMI oferece soluções inovadoras e eficientes para garantir o suprimento dos requisitos empresariais, permitindo a prática de ações que possam integrar todas as etapas de desenvolvimento por meio de níveis de maturidade, cuja principal função é assegurar com que a qualidade seja alcançada de maneira satisfatória.

Os testes mais realizados nas empresas envolvem os de unidade, que busca verificar a existência de falhas nos módulos (teste de caixa branca); integração, responsável por identificar falhas em diferentes unidades dos componentes do software; validação, analisa o processo de estruturação do software; e de sistema, que verifica os atributos voltados para o funcionamento do software.

Através da pesquisa com as empresas, foi possível compreender que apenas a empresa de grande porte possui uma área específica para a realização de testes, bem como mantem profissionais certificados neste processo.

Embora todas as empresas se dediquem a avaliação dos softwares, apenas as empresas de pequeno e grande porte realizam o devido planejamento e mantem uma documentação dos erros encontrados. A grande empresa demonstra uma preocupação em contratar profissionais que atuem no cargo de analista de testes.

Em relação às dificuldades em implantar os processos existentes, a pequena empresa considera que seu maior obstáculo é obter maior aceitação das interferências externas, enquanto a empresa de médio porte necessita de evolução na programação orientada a testes; e a empresa de grande porte relatou que enfrentou dificuldades por conta do modelo de documentação que utilizava nos projetos.

REFERÊNCIAS

ARANTES, D. de O. **Diagnóstico de um ambiente de Desenvolvimento de software segundo Práticas do CMMI**. 2007. 32 f. Monografia (Pós graduação) - Universidade Federal de Lavras de Minas Gerais. 2007.

BARBOSA, E. F et al. **Introdução ao teste de software**. In: Minicurso apresentado no SBES'2000 – XIV Simpósio Brasileiro de Engenharia de Software, João Pessoa, PB, 2000, p. 330-378.

BASILI, V. R; CALDIERA, G; CANTONE, G. A. A reference architecture for the componente factory. **Revista ACM**, jan, 1992.

CAROSIA, Jaciara Silva. **Levantamento da qualidade do processo de Software com foco em pequenas organizações** (2004) Disponível em: <http://www2.dem.inpe.br/ijar/Qualidade%20de%20Software/PDFs/publicacaoQualSW.pdf> Acesso em: 24 set. 2014

FALBO, Ricardo de Almeida. **Engenharia de Software** (2005) Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf> Acesso em: 25 nov. 2014

FERNANDES, A. A; TEIXEIRA, D. S. **Fábrica de software:** implantação e gestão de operações. São Paulo: Atlas, 2004.

LEFFINGWELL, Dean. **Calculating your return on investment from more effective requirements management** (2005) Disponível em: < http://www.silicontaiga.org/home.asp?artId=3394> Acesso em: 25 nov. 2014

LOZANO, F. **Testes: Ferramentas e Boas práticas:** Muito além do JUnit (2014) Disponível em: <http://www.devmedia.com.br/testes-ferramentas-e-boas-praticas-muito-alem-do-junit/8548> Acesso em: 25 nov. 2014

MARTINS, E. **Testes de Unidades e de Integração** (2009) Disponível em: <http://www.ic.unicamp.br/~eliane/Cursos/MC636-2009/Aula18-testes_unid_integracao.pdf> Acesso em: 25 nov. 2014

MAZIERO, Carlos Alberto. **Sistemas Operacionais** (2008) Disponível em: <http://www.ifba.edu.br/professores/flaviamsn/docs/so-cap01.pdf> Acesso em: 24 nov. 2014

.

MAZZOLA, V. M. **Engenharia de software:** conceitos básicos (2010) Disponível em: <http://jalvesnicacio.files.wordpress.com/2010/03/engenharia-de-software.pdf> Acesso em: 24 set. 2014

NOMURA, L. Definição e estabelecimento de processos de fábrica de software em uma organização de TI do setor (2008) Disponível em: <file:///D:/Downloads/2008_LuzNom_TeseDoutorado%20(1).pdf> Acesso em: 24 set. 2014

OCHNER, J; MENDES, R. D. **Guia para Implantação de Testes em Pequenas e Médias Empresas de Software** (2007) Disponível em: <http://ebts2007.cesar.org.br/Artigos/II-EBTS_GuiaImplanta%C3%A7ao_OUT-2007.pdf> Acesso em: 23 set. 2014

PFLEEGER, S. L. **Engenharia de Software: Teoria e Prática**. 2.ed. São Paulo: Prentice Hall, 004.

PRESSMAN, R. S. **Engenharia de Software**. Rio de Janeiro: McGraw-Hill, 2005.

ROCHA, A. R. C et al. **Qualidade de software:** Teoria e prática. Prentice Hall: São Paulo, 2001.

SEBRAE. **Critérios de classificação de empresas:** EI - ME - EPP. Disponível em: <http://www.sebrae-sc.com.br/leis/default.asp?vcdtexto=4154> Acesso em: 14 out. 2014

SILVA, F. B. **Proposta de um Framework para Fábricas de Software** (2005) Disponível em: <http://fabianabig.dominiotemporario.com/artigos/gerenciaprojetosfabricasoftware.pdf> Acesso em: 23 set. 2014

SILVA, F. C. da. **A Garantia da Qualidade de Software no Processo de Desenvolvimento** (2010) Disponível em: <http://www.artigonal.com/programacao-artigos/a-garantia-da-qualidade-de-software-no-processo-de-desenvolvimento-1836900.html> Acesso em: 23 set. 2014

THE STANDISH GROUP INTERNATIONAL. **Chaos manifesto 2013.** Disponível em: <http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf> Acesso em: 23 set. 2014

TIGRE, Paulo Bastos. **Computadores brasileiros:** indústria, tecnologia e dependência. Rio de Janeiro: Campus, 1984.

APÊNDICE A – Questionário aplicado com as empresas de pequeno, médio e grande porte.

1. Possui uma área dedicada à execução e suporte aos processos de testes? Quantos profissionais estão lotados nessa área? Quantos tem certificação em Testes de Software?

2. Quais as etapas do processo de testes de software (desde o planejamento do teste)?

3. Quais os tipos de testes efetuados?

4. Quantos tipos diferentes de testes são aplicados num mesmo projeto de desenvolvimento de software?

5. Quais os documentos utilizados para planejamento e execução dos testes?

6. Quais as condições mínimas para um teste de software ser considerado “aprovado”?

7. Qual o tempo médio planejado para testes, no cronograma original, e qual o tempo médio efetivamente dedicado aos testes no final do desenvolvimento?

8. Quais os softwares usados para apoiar as atividades de planejamento e de realização de testes?

9. O responsável final pelos testes tem qual nível hierárquico com a organização (diretor, gerente de setor, gerente de projeto, analista, testador etc.)?

10. Qual a quantidade (percentual) de resultados de testes “rejeitados” ou “não aprovados” desde o início deste ano?

11. Em que momentos, ou fases do desenvolvimento, os testes são realizados? Quais atividades de teste são executadas pelo próprio desenvolvedor? Quais são realizadas pelo analista de requisitos?
    Os testes unitários são realizados ao passo que o desenvolvedor implementa uma funcionalidade.

12. Qual o perfil desejado para a contratação de um analista de testes na sua organização?

13. Em que fases do processo de testes o cliente é envolvido? Qual documento registra esse envolvimento?

14. Quais foram as dificuldades em implantar o(s) processo(s) existente(s) e quais são as que ainda têm em implantar algum outro tipo de teste na empresa?