**UNIVERSIDADE FEDERAL DO PARÁ**
**INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

INÁCIO LEITE GORAYEB

# UM RELATO DE MELHORIA DO PROCESSO DE TESTE SOFTWARE APLICADO A UMA FÁBRICA DE SOFTWARE.

DM 47/2011

UFPA / ICEN / PPGCC
Campus Universitário do Guamá
Belém-Pará-Brasil
2011

**UNIVERSIDADE FEDERAL DO PARÁ**
**INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

INÁCIO LEITE GORAYEB

# UM RELATO DE MELHORIA DO PROCESSO DE TESTE SOFTWARE APLICADO A UMA FÁBRICA DE SOFTWARE.

DM 47/2011

UFPA / ICEN / PPGCC
Campus Universitário do Guamá
Belém-Pará-Brasil
2011

**UNIVERSIDADE FEDERAL DO PARÁ**
**INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

INÁCIO LEITE GORAYEB

# UM RELATO DE MELHORIA DO PROCESSO DE TESTE SOFTWARE APLICADO A UMA FÁBRICA DE SOFTWARE.

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Ciência da Computação da UFPA para a obtenção do Grau de Mestre em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.
Área de Concentração Engenharia de Software.
Orientador Prof. Dr. Sandro Ronaldo Bezerra Oliveira.

UFPA / ICEN / PPGCC

Campus Universitário do Guamá

Belém-Pará-Brasil

2011

Gorayeb, Inácio Leite

**Um Relato De Melhoria do Processo de Teste Software Aplicado a uma Fábrica de Software**, 2011. 169 páginas.

Dissertação (Mestrado)– Universidade Federal do Pará. Programa de Pós-Graduação em Ciência da Computação.

1. Processo de Desenvolvimento de Software 2. Qualidade de Software 3. Modelo de Qualidade 4. Engenharia de Software 5. Teste de Software. I. Universidade Federal do Pará. Instituto de Ciências Exatas e Naturais. Programa de Pós-Graduação em Ciência da Computação II-t

**UNIVERSIDADE FEDERAL DO PARÁ**
**INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

# UM RELATO DE MELHORIA DO PROCESSO DE TESTE DE SOFTWARE APLICADO A UMA FÁBRICA DE SOFTWARE.

INÁCIO LEITE GORAYEB

DISSERTAÇÃO DE MESTRADO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA A SER AVALIADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO PARÁ.

Data da aprovação: Belém-Pa. __-__-____

BANCA EXAMINADORA:

_____
Prof. Dr. Sandro Ronaldo Bezerra Oliveira
(ORIENTADOR – UFPA – Programa de Pós-Graduação em Ciência da Computação)

_____
Prof. Dr. Josivaldo de Souza Araújo
(UFPA – Faculdade de Computação)

_____
Prof. Dr. Dionne Cavalcante Monteiro
(UFPA – Programa de Pós-Graduação em Ciência da Computação)

# Agradecimentos

Primeiro a Deus que meu deu forças e ajudou a concluir este trabalho.

Aos meus pais, Inocêncio Gorayeb e Maria de Fátima Gorayeb e minha irmã Aína Gorayeb por todas oportunidades me proporcionadas ao longo da vida e por todo o incentivo, cobranças, e apoio dados.

A minha amada esposa Aline Gorayeb por estar sempre ao meu lado oferecendo seu amor, sua atenção, incentivando-me, criticando quando necessário e sempre apoiando e vivendo junto comigo meus sonhos.

Ao meu amigo e orientador Sandro Rolando Bezerra Oliveira pela generosidade de ter aceito o desafio desta orientação em um momento delicado, pela confiança e atenção dispensados para que este trabalho pudesse ser realizado e, acima de tudo, pela oportunidade proporcionada.

Agradeço também aos professores Josivaldo Araújo e Dionne Monteiro por aceitaram participar desta banca de mestrado.

Agradeço aos amigos e colegas de trabalho pela oportunidade de termos desenvolvido este trabalho com sucesso e nominalmente aos amigos Marcus Paulo e Harleu Abreu, por seus conhecimentos e conselhos sempre bem-vindos.

A todos aqueles que direta ou indiretamente contribuíram para a Realização deste trabalho.

"O insucesso é apenas uma
oportunidade para recomeçar
com mais inteligência "

(Henry Ford)

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Quadros

# Lista de Abreviaturas

| | |
|---|---|
| CVS | *Concurrent Version System* |
| PET | *Programa de Excelência Tecnológica* |
| SGF | *Sistema de Gestão de Fomento* |
| UML | *Unified Modeling Language* |
| SLA | *Service Level Agreement* |
| UML | *Unified Modeling Language* |
| BPMN | *Busines Process Model Notation* |
| VER | *Verificação* |
| VAL | *Validação* |
| TMM | *Test Maturiry Model* |
| TPI | *Test Process Improvment* |
| UCP | *Use Case Points* |

# Resumo

Qualidade de Software é a área da Engenharia de Software que se preocupa em observar a qualidade do processo para que produtos com qualidade possam ser produzidos e disponibilizados pelas organizações. É possível, então, que as organizações tenham como meta a implementação de um processo de Teste de Software.

O Processo de Teste de Software deve então planejar, projetar e executar procedimentos para garantir que o produto que está sendo gerado, está fazendo o que deve fazer e não está fazendo o que não deve fazer, tornando-o, assim, apto à utilização em ambiente de produção com comportamento e qualidade esperados.

Nesse contexto, este trabalho busca contribuir cientificamente apresentando um relato de experiência onde, a partir da análise de um processo onde foram extraídos pontos fracos, pontos fortes e oportunidades de melhorias e aplicação dos Modelos de melhorias de Processo TMM (*Test Maturity Model*) e MPS.BR, foi modelado um novo processo de Teste de Software com foco nos ativos de melhoria. Este processo foi avaliado em um ambiente real através de análise quantitativa e qualitativa a fim de verificar se estava atendendo às expectativas da organização.

**Palavras-Chave**: Engenharia de Software, Qualidade de Software, Melhoria Contínua, Teste de Software, Processo de Software.

# Abstract

Software Quality is the area of software engineering that is concerned with observe the quality of the process so that quality products can be produced and made available by organizations. It is possible, then, that the organizations have the goal of implementing a process of testing Software.

The Process of Software Testing must then plan, design and implement procedures to ensure that the product is being generated, is doing what they should do and not doing what you should not do, making The thus fit for use in production environment with behavior and

expected quality.

In this context, this paper seeks to contribute papers presented a report of an experience where, from the analysis of a process where they were extracted weaknesses, strengths and opportunities for improvements and application of process improvement model TMM (Test Maturity Model) and MPS.BR, a new process was modeled Software Testing with focus on improvement activities. This process was evaluated in a real environment through quantitative and qualitative analysis to verify if he was attending the expectations of the organization.

**Keywords:** Software Development Process, Software Development, Quality Models, Software Engineering, Software Process.

# Capítulo 1 – **Introdução**

A Engenharia de Software é uma das áreas da Ciência da Computação que possui como principal objetivo desenvolver um produto de software através de processos bem definidos a atender de maneira satisfatória o que foi solicitado.

Segundo Sommerville (2007), o software por sua própria natureza, é abstrato e intangível, o que torna o desenvolvimento de software uma tarefa não trivial de ser desenvolvida. Na década de 70, ocorreu um fenômeno que na literatura ficou conhecido como "Crise do software" (Koscianski, 2007), onde os softwares produzidos não atendiam as solicitações dos clientes, tornando-os confusos e aquém da capacidade dos hardwares.

Motivados pela "Crise do Software", muitos estudos diferentes foram desenvolvidos para apoiar e ajudar a resolver este cenário. Com isso, surgiu dentro da Engenharia de Software uma área conhecida como Qualidade de Software, que possui como objetivo possibilitar a produção de software cada vez mais adequado ao contexto para qual foi solicitado, por meio do acompanhamento do processo de produção do software, ou seja, o software deve estar em conformidade com os requisitos (Koscianski, 2007). Os resultados desta crise foram produtos de software entregues fora do prazo determinado, custos estipulados para o desenvolvimento ultrapassados e baixa qualidade dos mesmos.

Segundo Pressman (2007), a qualidade de software é dada por uma combinação complexa de fatores que variam de acordo com a aplicação em questão e as solicitações realizadas pelos clientes.

Outra ação de apoio para melhorar a qualidade dos software produzidos foi o investimento em Teste de Software, onde, segundo Hetzel (1987), Teste de Software é uma das atividades de verificação e validação cujo objetivo é avaliar uma característica ou recurso de um programa ou sistema, sendo considerada elemento crítico para a Garantia e o Controle da Qualidade do Software.

Com a qualidade em foco, várias normas e modelos de qualidades foram definidos, com o objetivo de incrementar a maturidade do processo que visam acompanhar o dia a dia da

organização. Alguns modelos envolvendo todas as funções e os níveis das organizações, objetivando a melhoria contínua da organização para que estas sejam capazes de satisfazer cada vez mais seus clientes. Alguns desses programas são voltados para a área de desenvolvimento de software, como MPS.Br (Softex, 2011), e outros diretamente para sub áreas da Engenharia de Software como o TMM, voltado para o Teste de Software.

Com base em programas de melhorias de processo de desenvolvimento e de teste de software esse trabalho mostra a aplicação de uma metodologia utilizada a partir de relato de experiência de melhoria de processo de teste de software, onde através de análises, entrevistas e auditorias, foi gerada uma nova versão do processo de teste de software que, posteriormente foi avaliado a partir de seus resultados.

## 1.1 Motivações

Com a busca por qualidade como diferencial de mercado e como fator determinante para o sucesso de uma organização, surge a motivação para a realização deste trabalho. Foi encontrado em uma organização um cenário propício para aplicar uma metodologia de melhorias de processo no processo de teste de software executado a partir de um projeto.

A organização encontrava dificuldade em validar o sistema com o cliente, que constantemente apresentava baixa qualidade de maneira a prejudicar a aceitação do produto. O processo de teste de software adotado não atendia as especificidades e necessidades do projeto, desta forma, foi proposta uma avaliação no processo de teste de software para que fossem agregadas melhorias que refletissem a realidade do projeto.

Apesar de existir várias opções disponíveis de modelos de processos de teste a serem utilizados pela organizações, características específicas inerentes ao projeto e à organização criaram a necessidade da modelagem de um processo específico para atender as demandas.

Além de contribuir para a organização, os resultados deste trabalho: a metodologia de análise e definição utilizada; o processo de teste de software modelado, podem ser utilizados em cenários com necessidades parecidas com o da fábrica de software utilizada para o relato da experiência.

## 1.2 Objetivo

De acordo com os pontos citados, o objetivo deste trabalho é realizar uma avaliação no processo de teste de software utilizado pela organização, onde, através de uma metodologia, tentou-se promover melhorias utilizando como parâmetros recomendações de

modelos de referência de melhorias de processos. Após as melhorias, realizar avaliações quantitativas e qualitativas a fim de verificar se o processo de teste modelado atingiu os objetivos da organização e do projeto, e se a metodologia de melhoria adotada obteve sucesso.

### 1.2.1 **Objetivos específicos**

- Apresentar a importância de processos de softwares e modelos de qualidade organizacionais para o dia a dia das organizações;
- Apresentar a relevância do Teste de Software e a adoção de um Processo de Teste de software na busca pela qualidade dos produtos;
- Avaliar e melhorar um processo de Teste de Software através de pontos fracos, fortes e oportunidades de melhorias;
- Definir um processo de teste de software que atenda aos objetivos da organização, do projeto e dos modelos de referência MPS.BR e TMM;
- Desenvolver uma metodologia de avaliação para o processo de teste de software modelado;
- Avaliar as melhorias do processo modelado quantitativamente, através da adoção de GQM (*Goal Question Metric*), e qualitativamente, a partir de entrevistas com os participantes do processo.

## 1.3 Contribuições

De acordo com os objetivos descritos na seção anterior, as seguintes contribuições são apresentadas nesta dissertação:

- Descrição de uma metodologia de melhoria de processo aplicado a um processo de teste de software;
- Descrição da aplicação de modelos de melhorias de processo em um processo de teste de software, apresentando um processo de teste de software, que foi modelado de acordo com as exigências do MPS.BR com foco nos processos de Verificação e Validação;
- Descrição da modelagem de um processo de teste de software, utilizando uma linguagem de modelagem de processo BPMN para descrever o processo de teste gerado após as melhorias propostas na avaliação;

- Descrição da análise do processo através de uma avaliação quantitativa, onde foi realizada uma avaliação quantitativa utilizando o GQM;
- Descrição da análise do processo através de uma avaliação qualitativa, onde foi realizada uma avaliação qualitativa através de entrevistas com os participantes do processo.

## 1.4 Metodologia

Esta seção descreve como decorreu a metodologia para o desenvolvimento deste trabalho. Este trabalho foi dividido nas seguintes etapas:

**Levantamento inicial do cenário**

- Pesquisa e estudo em trabalhos relacionados, artigos, documentos, livros e *sites* da área de Engenharia de Software e Teste de Software, mostrando uma visão geral de um: processo de desenvolvimento de software, processo de teste de software, modelos de melhorias, assim como suas limitações;
- Estudo nos modelos e normas de qualidade para processos de software a fim de realizar melhorias em um processo de teste de software;
- Estudo em técnicas de métricas como GQM;
- Pesquisa e estudo na área de Qualidade de Software, que serviram como base para avaliar e promover melhorias no processo de desenvolvimento;
- Estudo aprofundado de trabalhos na área de Teste de Software, Qualidade de Software e Engenharia de Software que serviram de fundamentação para elaboração do estado da arte desse trabalho

**Etapa do levantamento inicial do Processo de Teste de Software**

- Levantamento de informações sobre o processo de Teste de software utilizado na Fábrica de software a fim de identificar pontos fortes, fracos e oportunidades de melhorias;
- Levantamento dos problemas, dificuldades e objetivos da organização, a fim de avaliar o problema na qualidade final do produto de software.
- Estudo de normas e modelos de qualidade, com o objetivo de obter boas práticas para aplicação da melhoria processo de teste de software.

**Etapa de formas de avaliação**

- Estudo em notação de processos, através do BPMN, a fim de adotar uma linguagem especializada em modelagem de processo que fosse de fácil compreensão e de simples descrição;
- Estudo de formas para avaliação quantitativa, afim de avaliar se o processo de teste de software modelado atente aos objetivos da organização, do projeto e dos interessados;
- Estudo em formas para avaliação qualitativa, a fim de avaliar se o processo de teste de software modelado atende os objetivos da organização, do projeto e dos interessados, além de verificar o grau de satisfação dos envolvidos diretamente;
- Avaliar as mudanças, a fim de verificar se as mesmas atingiram os objetivos.

**Etapa de melhorias**

- Verificar os pontos fracos e Oportunidade de melhorias;
- Prover um Plano de Ação para realizar melhorias no processo de Teste de Software.

Com base na literatura especializada (Silva, 2001), existem diversas formas para classificar a pesquisa realizada nesta dissertação, que podem ser: quanto a **Natureza**, onde, adotou-se Pesquisa Aplicada, pois tem como objetivo gerar conhecimento para a aplicação prática dirigidos à solução de problemas específicos; quanto a **Abordagem do Problema**, onde usou-se da Pesquisa Quantitativa, que considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificar e analisar, enquanto que a Pesquisa Qualitativa, trata de tudo o que não pode ser traduzido em números. A interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa, não requer o uso de métodos e técnicas estatísticas.

Já quanto aos **Objetivos**, a Pesquisa Exploratória, pois tem o objetivo de proporcionar maior familiaridade com o problema, através de uma pesquisa bibliográfica, entrevistas com pessoas que tiveram experiências práticas com a melhoria do processo de Teste de Software, e a Pesquisa Descritiva, também foi adotada pois tem o objetivo de descrever as características de determinada população ou fenômeno ou o estabelecimento de relações entre variáveis. Envolve o uso de técnicas padronizadas de coleta de dados: questionário e observação sistemática.

Quanto aos **Procedimentos Técnicos**, usou-se da Pesquisa Bibliográfica, elaborada através da coleta de materiais publicados como artigos de periódicos e materiais disponibilizados na Internet e através de Pesquisa Experimental, pois quando se determina um objeto de estudo, selecionam-se as variáveis que seriam capazes de influenciar, definem-se as formas de controle e de observação dos efeitos que a variável produz no objeto.

## 1.5 Organização da Dissertação

Esta dissertação está organizada em seis capítulos, a saber:

- CAPÍTULO 2 – apresenta o resultado do estudo bibliográfico abrangendo processo de software, modelos para definição de processo de software, normas de qualidade e modelos de Processo de Desenvolvimento de Software de uma maneira geral;
- CAPÍTULO 3 – apresenta o resultado bibliográfico e um detalhamento sobre Teste de Software, mostrando a importância de um processo de Teste de software, os modelos de Teste, os Tipos de Teste;
- CAPÍTULO 4 – apresenta a contextualização do cenário do projeto, do cliente e do fornecedor. Após a apresentação dos cenários é mostrado o processo de desenvolvimento da organização e o processo de teste versão 1 e posteriormente sua avaliação, tendo como resultado pontos fortes, fracos e oportunidades de melhorias. Com base nesse resultado, em modelos de melhorias de processos e, com base nos objetivos da organização, do projeto e de interessados foi apresentado o Processo de Teste de Software Versão 2, que foi descrito com a utilização da notação BPMN;
- CAPÍTULO 5 – apresenta a avaliação do processo de Teste de Software Versão 2, onde é abordada a metodologia utilizada em sua avaliação, o relato do estudo de caso, a avaliação quantitativa através da definição e análise de métricas, a avaliação qualitativa onde foi realizada aplicação de questionários para os envolvidos e seus resultados são expostos e analisados;
- CAPÍTULO 6 – apresenta as conclusões deste trabalho onde serão apresentadas as contribuições deste trabalho, as limitações do processo desenvolvido, os resultados obtidos com a adoção do processo, e os trabalhos futuros pretendidos.

# Capítulo 2 - **Definição e Melhoria de Processo de Software**

As mudanças que estão ocorrendo nos ambientes de negócios têm motivado as empresas a modificar estruturas organizacionais e processos produtivos, saindo da visão tradicional baseada em áreas funcionais em direção a redes de processos centralizados no cliente. A competitividade depende, cada vez mais, do estabelecimento de conexões nestas redes, criando elos essenciais nas cadeias produtivas. Alcançar competitividade pela qualidade, para as empresas de software, implica tanto na melhoria de qualidade dos produtos de software e serviços correlatos, como dos processos de produção e distribuição de software.

Desta forma, assim como para outros setores, qualidade é fator critico de sucesso para a indústria de software. Para que se tenha um setor de software competitivo nacional e internacionalmente, é essencial que os empreendedores do setor coloquem a eficiência e a eficácia dos seus processos em foco nas empresas, visando à oferta de produtos de software e serviços correlatos conforme padrões internacionais de qualidade.

Este capitulo abordará conceitos e definições sobre processo de software, modelos de definição de processo de software, ferramentas para automação de definição de processo de software, programas de melhorias de processo de software, normas e modelos para definição e melhoria de processo de software e por fim fará um paralelo entre qualidade do processo x qualidade do produto.

## 2.1 Conceitos Iniciais

Humphrey (apud Oliveira, 2006) define processo de software como o conjunto de tarefas de Engenharia de Software [1]necessárias para transformar os requisitos dos usuários em software. Na definição de um processo de software devem ser consideradas as seguintes informações: atividades a serem realizadas; recursos utilizados; artefatos consumidos e

[1] **Engenharia de Software** é uma área do conhecimento da computação voltada para a especificação, desenvolvimento e manutenção de sistemas de software aplicando tecnologias e práticas de gerência de projetos e outras disciplinas, objetivando organização, produtividade e qualidade

gerados; procedimentos adotados; paradigma e tecnologia adotados; e o modelo de ciclo de vida utilizado (Falbo, 1999) (Pfleeger, 2004). Posteriormente um detalhamento será dado sobre esta visão.

Processos de software são as diversas fases necessárias para produzir e manter um produto de software. Requerem a organização lógica de diversas atividades técnicas e gerenciais envolvendo agentes, métodos, ferramentas, artefatos e restrições que possibilitam disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software. É através da descrição formal de um processo de software que permite que o mesmo seja analisado, compreendido e automatizado (executado). Sendo assim, a modelagem e a execução de processo de software é de fundamental importância para o aumento da qualidade do produto de software e têm sido estudadas pela comunidade de Engenharia de Software com o intuito de prover ferramentas que a suportam.

A tecnologia do processo de software ou simplesmente processo de software (Reis, 1999) é um dos maiores alvo de estudos, pesquisas e práticas da Engenharia de Software. Para que um processo de software sofra automação, verificação, aperfeiçoamento e reutilização a tecnologia baseia-se no conceito de que o processo de software seja descrito formalmente.

A origem do processo de software está relacionada à primeira origem de um ciclo de vida para o software, ou seja, uma definição de um conjunto de estágios que forneça subsídios para o acompanhamento do desenvolvimento deste processo de software.

Informalmente, o processo de software pode ser compreendido como um conjunto de todas as atividades necessárias para transformar os requisitos do usuário em software. Um processo de software é formado por um conjunto de passos de processo parcialmente ordenados, relacionados com conjuntos de artefatos, pessoas, recursos, estruturas organizacionais e restrições e tem como objetivo produzir e manter os produtos de software finais requeridos (Reis, 1999).

Os passos de um processo são atividades ou tarefas. Atividades ou tarefas definem "como as coisas são feitas" ou "o que será feito" e para isso incorporam procedimentos, regras, políticas, agentes, recursos, papéis, artefatos (consumidos e produzidos). Cada atividade deve definir claramente seu início e final. Normalmente, atividades são organizadas em uma rede de duas dimensões: horizontal e vertical. A dimensão horizontal define os relacionamentos entre as atividades e a dimensão vertical decompõe a atividade em diversos níveis, ou seja, uma atividade composta por diversas outras sub-atividades. Exemplo de sub-atividades seria a decomposição da atividade gerenciamento de projeto nas sub-atividades:

estimativas de tamanho, estimativas de custo, controle e acompanhamento como mostra a Figura 2-1.

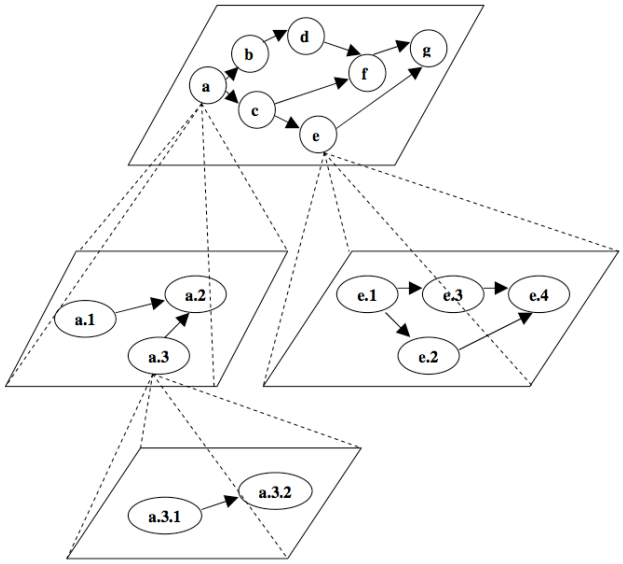


Figura 2-1 – Rede de Tarefas (Reis, 1999).

Uma atividade aloca recursos (por exemplo, máquinas e orçamento), é escalonada, monitorada e atribuída a desenvolvedores (agentes), que podem utilizar ferramentas para executá-la (Reis, 1999). Uma atividade também pode ser executada somente por ferramentas automatizadas, sem intervenção humana. Toda atividade possui uma descrição, a qual pode especificar os artefatos necessários, as relações de dependência com outras atividades, a data de início e fim planejadas, os recursos a serem alocados e os agentes responsáveis pela mesma.

Agentes, atores ou ferramentas (atividades automatizadas), são as entidades que executam atividades por intermédio de um papel. Os agentes podem estar organizados em cargos (papéis) e diferentes agentes terão percepções acerca do que acontece durante o processo de software. Um agente Gerente, por exemplo, perceberá os aspectos de controle, alocação de recursos e cronogramas para atividades, enquanto um Analista de Teste, por exemplo, perceberá as atividades relacionadas a controlar a qualidade do produto gerado.

Um artefato é um produto criado ou modificado durante um processo. Tal produto é resultado de uma atividade e pode ser utilizado posteriormente como matéria-prima para a mesma ou para outra atividade a fim de gerar novos produtos. Desta forma, uma atividade

pode consumir produtos (de entrada) e gerar novos produtos (de saída). Os produtos são frequentemente persistentes e possuem versões (Reis 1999). Em um contexto de projeto de software, alguns artefatos existem apenas na forma de um arquivo lógico (arquivo executável, por exemplo).

Geralmente, artefatos são persistentes e as possíveis modificações sofridas por um artefato podem produzir versões de um mesmo artefato (visionados). Artefatos podem ser representados como um agregado de sub-artefatos (módulos de código-fonte) ou artefatos mais complexos podem ser decompostos em sub-artefatos (Projeto Procedimental). Exemplos de artefatos são, entre outros: código-fonte, código executável, manual de padrões, relatório de resultados, documento de requisitos, plano de trabalho.

Porém, não existe um processo de software genérico de maneira que possa ser aplicado a vários projetos, uma vez havendo peculiaridades nas variações, nas políticas, procedimentos organizacionais, métodos e estratégias de aquisição, tamanho e complexidade do projeto, requisitos e métodos de desenvolvimento do sistema em cada um deles. Portanto, na definição de um processo deve-se considerar tecnologias envolvidas, ao tipo de software em questão, ao domínio de aplicação, ao grau de maturidade (ou capacitação) da equipe em engenharia de software, às características próprias da organização, às características do projeto e da equipe (Rocha, apud Oliveira, 2006).

### 2.1.1 **Processo Padrão**

Em uma organização ou grupo de desenvolvimento multi-organizacional, diversos projetos podem coexistir possuindo características específicas. Porém, existe um conjunto de elementos fundamentais que deseja-se que sejam incorporados em qualquer processo definido (Oliveira, 2006). A (ISO/IEC, 2008) define o conjunto destes elementos fundamentais como o processo padrão, ou seja, o processo básico que guia o processo comum na organização. Desta forma, um processo padrão define uma estrutura única a ser seguida por todas as equipes envolvidas em um projeto de software, independente das características do software a ser desenvolvido. (Humphrey, 1999) define um conjunto de razões para a definição de um processo padrão são eles:

- Redução de treinamentos, revisões e suporte a ferramentas;
- As experiências adquiridas nos projetos são incorporadas ao processo padrão e contribuem para melhorias em todos os processos definidos;

- A utilização de padrões de processo, fornecendo as bases para medições de processos e qualidade;
- Economia de tempo e esforço em definir novos processos adequados a projetos;

Atualmente, há uma tendência à utilização de processos padrões na definição de processos. A norma ISO/IEC 12207 (ISO/IEC, 2008), o SPICE (ISO/IEC, 2008), o CMM (Paulk 1993), CMMI (Chrissis, 2003) e MPS.BR (Softex, 2011), definem um processo padrão como um ponto base a partir do qual um processo especializado poderá ser obtido de acordo com as características de um projeto de software específico. Tendo em vista a particularidade de cada projeto o processo de desenvolvimento de software padrão da organização deve ser adaptado (especializado) de acordo com as particularidades de cada projeto.

## 2.2 Modelos para Definição de Processo de Software

À medida que aumenta a preocupação com a qualidade de software, aumenta a preocupação das organizações em seguir as orientações de modelos e os padrões de qualidade de processo tais como ISO 9000-3 (ISO, 1991 apud Oliveira, 2006) CMMI (Chrissis, 2003), ISO/IEC 12207 (ISO/IEC, 2008), SPICE (ISO/IEC, 2008), e MPS.BR (Softex, 2011). Porém, a definição de processo de software não é uma tarefa trivial e envolve variáveis complexas e ainda no mercado possuem poucos profissionais qualificados para desempenhar esta tarefa.

Em uma organização, diferentes processos podem coexistir, adequados a diferentes projetos. Para organizar e disciplinar o desenvolvimento de software, é importante determinar as atividades fundamentais que deverão estar presentes em qualquer processo definido. A definição de um processo padrão estabelece uma estrutura comum a ser utilizada pela organização nos seus projetos de software e constitui a base para a definição de todos os processos (Rocha, 2001). Portanto, é definido um processo padrão que servirá de base para as futuras instâncias de processo de software que serão adequadas a projetos específicos com características também específicas.

Segundo Pressman (2007), um projeto é a instância de um processo, com objetivos e restrições específicos. Pode-se dizer que um projeto é um esforço para desenvolver um produto de software, ou seja, envolve uma estrutura organizacional, prazos, orçamentos, recursos e um processo de desenvolvimento.

Rocha (2001), apresenta um Modelo para Definição de Processo de Software onde, este modelo baseia-se na experiência de definição de processo de software para diferentes

domínios de aplicação e usando diferentes tecnologias de desenvolvimento de software. O modelo consiste em três etapas ilustradas na Figura 2-2. A primeira etapa é a Definição do Processo Padrão, onde pode ser realizada tendo como base a norma ISO/IEC 12207, além de poder considerar modelos de maturidade como o CMMI e/ou MPS.BR.

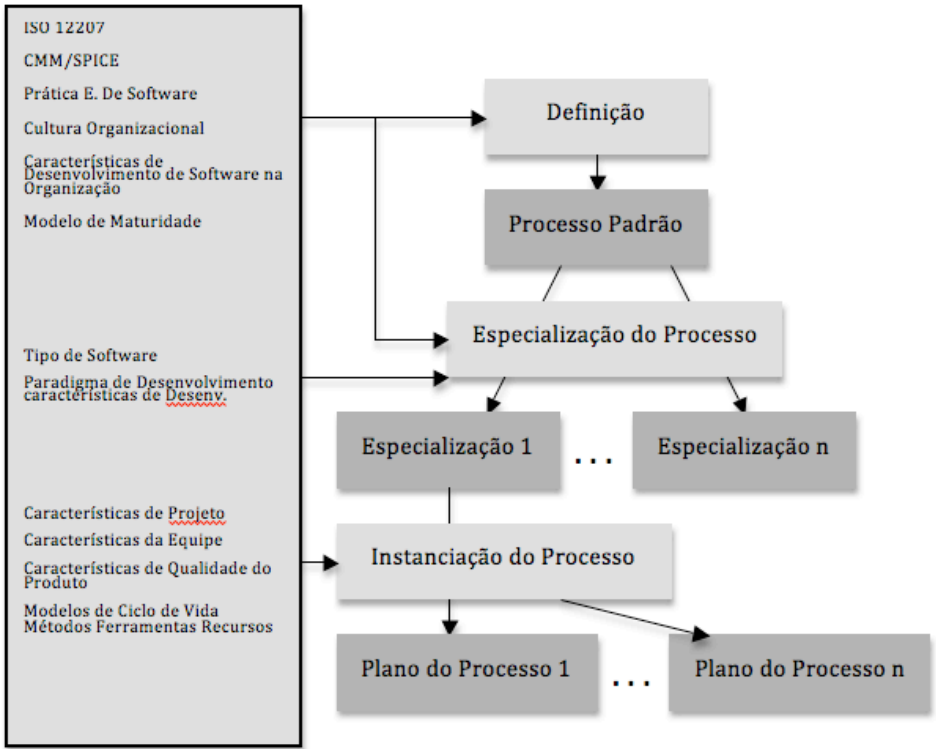


Figura 2-2 – Modelo para a definição de processos de softwares (Rocha, apud Oliveira, 2006).

Já a segunda etapa do modelo de definição para processo de software é a Especialização do Processo Padrão, onde deve ser levado em consideração as peculiaridades de cada projeto respeitando características e exigências diferentes que cada software possui, por exemplo, tipo de software (missão critica, *online*, sistemas de informação, etc.), paradigma de desenvolvimento utilizado (orientação a objetos, Lógico, etc.), nível de serviço (Oliveira, 2006). Com isso, nesta segunda etapa pode-se adicionar atividades ou modificar as existentes de acordo com a necessidade de cada software.

Segundo Oliveira, 2006, o desenvolvimento de uma instância de processo de software devidamente adaptada às particularidades do projeto é uma tarefa difícil, em virtude de informações que devam ser agregadas e das alterações que devem ser feitas no processo padrão da organização. Ainda segundo Oliveira, 2006, a dificuldade de adaptação não tem sido resolvida, pois as descrições de processo de software são representadas em um alto nível de abstração de forma a assumir diferentes variantes.

Algumas ações para resolução do problema de instanciação de processo de software são notadas, o governo alemão por exemplo através do uso de ferramentas oferece meios explícitos de adaptação do processo padrão para projetos específicos, como por exemplo a utilização do padrão "Das Vorgehensmodell" (V-Modell).

## 2.3 Experiência de uso automatizado à definição de Processo de Software

Muito tem se discutido na literatura sobre as propriedades dos ambientes de desenvolvimento de software, no entanto, percebe-se ao longo da execução do processo, a partir desta tecnologia, que sua implementação nem sempre satisfaz as necessidades das organizações ou dos projetos desenvolvidos por estas. Isto se deve ao fato de que os responsáveis pela definição do processo não dispõem de um guia contendo as suas reais necessidades de execução, indicando as melhores práticas a serem instanciadas a partir de um processo padrão (Oliveira, 2006).

Neste contexto, várias tentativas e experiências de automatização à definição de processo de software podem ser encontradas na literatura onde nem todas as atividades que perfazem o ciclo de vida de um processo de software (Definição, Simulação, Execução e Avaliação) são atendidas por completo.

O V-Modell apresentado na subseção anterior, possui meios para ser adaptado para projetos específicos, entretanto sua adaptação é limitada à remoção de elementos do processo padrão, que acaba por diminuir seu poder de customização. Para apoio a esse modelo de definição de processo de software a ferramenta ProcePT (Process Programming and Testing) (Welzel apud Oliveira, 2006) apresenta uma automatização da adaptação do V-Model para diferentes projetos através da deleção de atividades e produtos do processo.

Outras abordagens de automatização à definição de processo de software têm tido bastante destaque. A ferramenta AdaptPro (Berger, 2003) fornece ao gerente de projetos o conhecimento sobre instanciação de processo de software acumulado pela organização de software em projetos anteriores.

Já o projeto TABA propõe a construção de uma estação de trabalho para o desenvolvimento de software, configurável, para atender às particularidades de domínios de aplicação e projetos específicos. As principais ferramentas da estação TABA são ASSIST-PRO (Falbo apud Oliveira, 2006) e DEF-PRO (Machado, 2000). O DEF-PRO considera a automação da definição de processo a partir de um processo padrão para uma organização, definido de acordo com a ISO/IEC 12207, nos modelos de maturidade (CMMI e SPICE). Já o

ASSIS-PRO considera a definição do processo para projetos específicos, com base em um assistente inteligente que fornece diretrizes, sugestões e opções para aumentar a qualidade e a produtividade nesta tarefa.

O ImPProS (Implementação Progressiva de Processo de Software), consiste em uma ferramenta de automação e melhoria de processo de software desenvolvido pelo CIN/UFPE (Centro de Informática da Universidade Federal de Pernambuco), baseando-se em um meta-modelo de processo de software composto de componentes e dos relacionamentos entre esses que são oriundos do mapeamento de algumas normas e modelos de qualidade para processo de software (CMMI, SPICE – ISO 15504, ISO 12207, etc) (Oliveira, 2006). O objetivo do meta-modelo é determinar uma terminologia única para a definição de processo de software no ImPProS. Um diferencial do ImPPros que acaba ajudando na resolução da dificuldade de adequação de processos da organização é que o ambiente, contempla em sua composição todos os componentes de um processo de software (processo, atividades, artefatos, etc.) de vários modelos/normas de maneira a permitir que o usuário possa escolher de onde partirá seu meta-modelo.

Outra experiência de automação à definição de processo de Software automatizado por parte do laboratório LABES/UFPA é o ambiente WebAPSEE, que consiste em um PSEE[2] (*Process-Centered Software Engineering Enviroment*) para gestão de processos baseado em Software Livre onde sua primeira versão foi construída entre 2004 e 2005 com esforço de cooperação entre instituições acadêmico-científicas e um parceiro industrial (Regional Belém do Serviço Federal de Processamento de Dados – SERPRO). Neste ambiente foi criada uma linguagem visual (WebAPSEE-PML – *Process Modeling Language*) e formal com abordagem de processo orientado a atividade, usada para modelagem de processo no ambiente (LIMA, 2006).

Segundo (LIMA, 2006) WebAPSEE possui três componentes: atividades, conexões e artefatos. As atividades são ações realizadas por atores participantes do processo, já as conexões determinam as relações temporais e de sincronização entre atividades e os artefatos é a denominação genérica para referências de itens de software contidas em sistemas de controle de versão que são usados nos processos como mostra a Figura 2-3.

---

[2] PSEE - Ambientes de automação de processo de software que possibilitam a coordenação de atividades de equipes de desenvolvimento de software, acompanhamento dos prazos e utilização de recursos, além de facilitar a reutilização de boas práticas gerenciais a partir de diferentes projetos já concluídos.

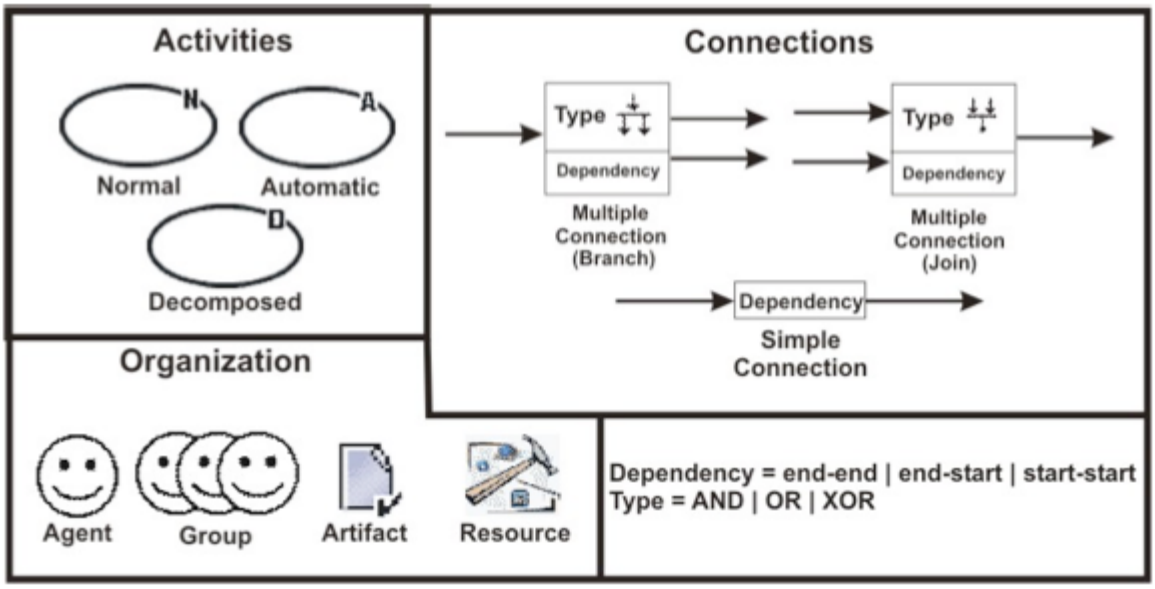


Figura 2-3 – Representação gráfica para as construções WebAPSEE-PML (LIMA, 2006).

## 2.4 Melhoria Contínua de Processo de Software

Com o crescimento da utilização de automação na vida moderna os softwares por sua vez sofreram um considerável crescimento tanto no tamanho quanto na complexidade de maneira a serem considerados vitais em diversos ramos da sociedade. Com isso, a preocupação com qualidade do produto, satisfação do cliente e custos do projeto aumentam.

Juntamente com as preocupações, os problemas enfrentados durante o desenvolvimento também aumentam, segundo (Humphrey apud Oliveira, 2006) a produtividade das equipes diminui à medida que crescem o tamanho e a complexidade dos sistemas, e isso ocorre devido à baixa qualidade dos sub-produtos que compõem o produto final. Além das preocupações e problemas citados, ainda existe a competitividade entre as produtoras de software que tornou a qualidade do produto não mais um diferencial e sim um requisito básico para a sobrevivência no mercado.

Com a qualidade como requisito básico, várias normas e modelos de qualidade foram definidos onde pode-se destacar como principais: Norma ISO/IEC (ISO/IEC, 2008), CMM (*Capability Maturity Model*) (Paul, 1993), SPICE (*Software Process Improvment and Capability Determination*) e os mais atuais ISO/IEC 15504 (ISO/IEC 2008), CMMI (*Capability Maturity Model Integration*) (CMMI, 2011) e, por fim e não menos importante o MPB.BR (Melhoria de Processo de Software Brasileiro) (Softex, 2011). Porém, para auxiliar na melhoria e definição do processo de software e alcançar níveis de qualidades mais altos torna-se necessário que melhore-se também cada etapa do ciclo de desenvolvimento.

Um exemplo de modelo de melhoria contínua de processo é o IDEAL, que originalmente consiste em um modelo de ciclo de vida para a melhoria do processo de software baseada no CMM (Paulk, apud Oliveira, 2006) e por esta razão usou termos da melhoria do processo. O modelo fornece um guia para a melhoria contínua das características de um processo organizacional, ou seja, o modelo fornece um guia disciplinado de engenharia para melhoria contínua focando no gerenciamento do programa de melhoria e estabelecendo base para uma estratégia de melhoria a longo prazo. O IDEAL é divido em cinco fases: *Initiating, Diagnosing, Establishing, Acting e Learning* como mostra a Figura 2-4 a seguir.

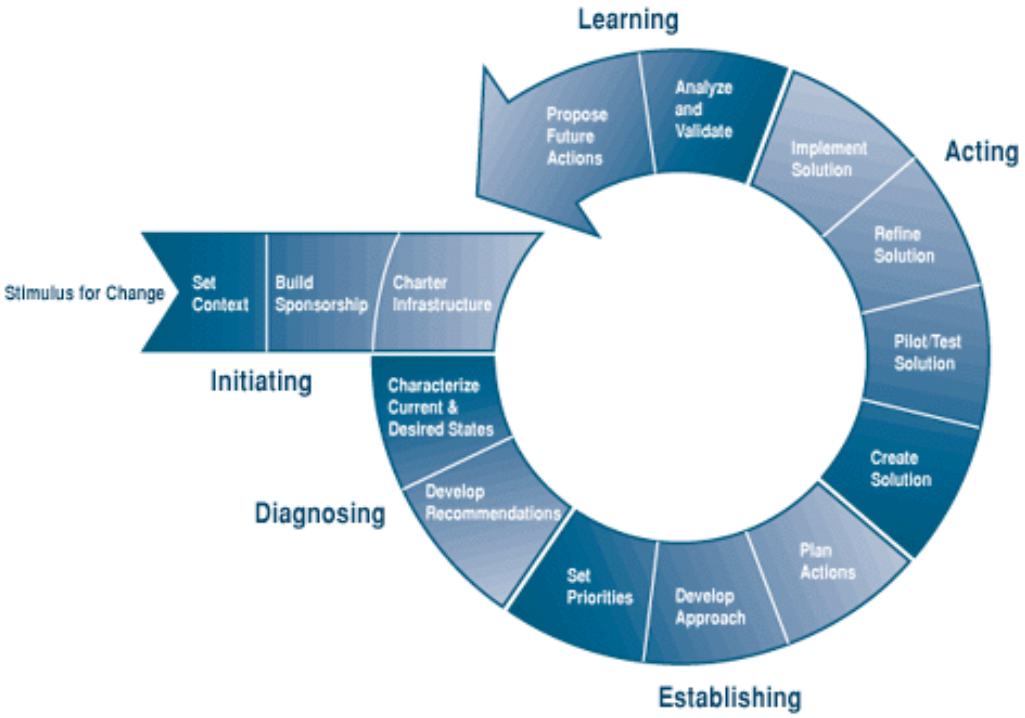


Figura 2-4 – Modelo IDAL (Mcfeeley, apud Oliveira, 2006).

A fase de *Initiating* é onde a infra-estrutura inicial da melhoria é estabelecida, os papéis e as responsabilidades são definidos e os recursos iniciais são atribuídos. A fase possui seu enfoque no estímulo para a melhoria do processo de software, na definição do contexto e do patrocinador e no estabelecimento da infra-estrutura inicial para suporte e melhoria.

A fase de *Diagnosing* possui como objetivo desenvolver um entendimento mais completo do trabalho de melhoria onde as estratégias de estado atual e estado futuro desejado são definidas. Esses estados da organização são utilizados para desenvolver uma abordagem para a prática de melhoria do negócio.

A fase de *Establishing* possui como intuito o desenvolvimento de um plano de trabalho detalhado onde as prioridades de quais práticas organizacionais serão melhoradas e

ajustadas para refletir as recomendações feitas nas fase *Diagnosing* tendo como finalidade de colocar prioridades para as alterações, desenvolver uma estratégia para realização do trabalho, identificar recursos disponíveis, e desenvolver um plano de implementação detalhado.

Já a fase de *Acting*, serve para ajudar a organização a implementar o trabalho realizado nas três fases anteriores (*Initiating*, *Diagnosing* e *Establishing*), tendo como foco a criação de uma solução que atenda as necessidades organizacionais identificadas, testar a solução criada através de um projeto piloto e implementar a solução em toda a organização.

Por fim, a fase de *Learning*, onde a experiência obtida na execução do modelo é revista para determinar se os esforços conseguiram atingir os objetivos pretendidos e como a organização pode executar a mudança mais eficaz e/ou eficientemente no futuro. Para isso, as lições são coletadas, analisadas, documentadas, onde as necessidades identificadas na fase *Initiating* são revistas para verificar seu atendimento.

Para que a organização tenha a real noção de como anda seu processo de desenvolvimento de software torna-se indispensável a utilização de métricas. Com isso, introduz-se à engenharia de desenvolvimento as medições que possuem como objetivo:

- Melhorar o entendimento sobre o processo, produto, recursos e ambiente de desenvolvimento e, assim, estabelecer bases para a comparação entre as medições;
- Avaliar o andamento do projeto comparando com dados planejados;
- Fazer previsões sobre o futuro andamento do projeto com base em comportamentos passados;
- Promover melhorias identificando falhas, ineficiências e outras oportunidades para melhorar a qualidade do produto e o desenvolvimento do processo.

## 2.5 Normas e Modelos para Definição e Melhoria de Processo de Software

Como discutido na seção anterior, diversos padrões e modelos têm sido desenvolvidos e mudanças vêm ocorrendo nos ambientes de negócios motivando empresas a modificar estruturas organizacionais e processos produtivos, saindo da visão tradicional baseada em áreas funcionais em direção a redes de processos centrados no cliente. Com maior preocupação com qualidade de software, as organizações passaram a adotar padrões internacionais e modelos na definição de processo de software, buscando assim melhorar a qualidade de seus produtos, aumentar a produtividade de suas equipes, e reduzir os custos e os riscos associados com o desenvolvimento de software.

A abordagem de utilização de padrões internacionais torna-se interessante no atual contexto de globalização da economia mundial e, também, devido aos altos custos associados ao desenvolvimento de padrões particulares a cada organização (ISO/IEC, 2008). Nesta seção serão abordadas as normas ISO/IEC 12207, ISO/IEC 15504, os modelos de maturidade CMM e CMMI, e o modelo de referência MPS.BR.

### 2.5.1 **Norma ISO/IEC 12207 – Tecnologia de Informação – Processos de Ciclo de Vida de Software**

A Norma ISO/IEC 12207 (ISO/IEC, 2008) possui como objetivo estabelecer uma estrutura comum para todos os processos de ciclo de vida de software, com terminologia bem definida, para ser utilizada por desenvolvedores para construir, gerenciar e melhorar o software. A estrutura contém processos, atividades e tarefas a serem aplicados durante a aquisição, fornecimento, desenvolvimento, manutenção e operação de produtos de software, e que devem ser adaptados ao contexto e características de cada projeto. A adaptação consiste basicamente na exclusão de processos e tarefas não aplicáveis. Com o objetivo de evitar conflitos com procedimentos já adotados pela organização, o processo de adaptação deverá estabelecer uma compatibilidade com políticas e normas já existentes na organização (ISO/IEC, 2008) (Rocha, 2001).

A Norma ISO/IEC 12207 apesar de descrever a arquitetura dos processos de ciclo de vida de software, não especifica detalhes de como implementar ou executar as atividades e tarefas incluídas nos processos. Outra característica da norma é deixar a encargo do usuário a definição do conteúdo e do formato da documentação a ser produzida. Além das características citadas anteriormente, a norma também não define um modelo de ciclo de vida a ser utilizado, nem um método de desenvolvimento de software a ser aplicado. Tal escolha será baseada nas características do projeto e da equipe envolvida.

Para a Norma ISO/IEC 12207 os processos de ciclo de vida são agrupados em três grandes classes como mostrado na Figura 2-5, sendo que cada processo de ciclo de vida é dividido em um conjunto de atividades e cada atividades em tarefas.

A primeira grande classe são os Processos Fundamentais que executam as principais funções durante o ciclo de vida do software onde são compostos de cinco processos: aquisição, fornecimento, desenvolvimento, operação e manutenção.

Já a segunda é composta pelos Processos de Apoio que auxiliam outros processos na execução de funções especializadas, contribuindo para o sucesso e qualidade do projeto de

software e são compostos por oito processos: documentação, gerência de configuração, garantia de qualidade, verificação e validação, revisão conjunta, auditoria e resolução de problemas.

E a terceira grande classe é composta pelos Processos Organizacionais e constituem um conjunto de quatro processos responsáveis pelo gerenciamento e suporte de todo o ambiente de desenvolvimento, sendo empregados normalmente, fora de domínios ou contratos específicos e são compostos de quatro processos: gerência, infra-estrutura, melhoria e treinamento.

<table>
<tr><td colspan="2">Processos Fundamentais</td><td colspan="2">Processos de Apoio</td><td rowspan="14">Processo de Adaptação</td></tr>
<tr><td colspan="2">Aquisição</td><td colspan="2">Documentação</td></tr>
<tr><td colspan="2">Fornecimento</td><td colspan="2">Gerência de Configuração</td></tr>
<tr><td rowspan="9">Desenvolvimento</td><td rowspan="4">Operação</td><td colspan="2">Garantia da Qualidade</td></tr>
<tr><td colspan="2">Verificação</td></tr>
<tr><td colspan="2">Validação</td></tr>
<tr><td colspan="2">Revisão Conjunta</td></tr>
<tr><td rowspan="5">Manutenção</td><td colspan="2">Auditoria</td></tr>
<tr><td colspan="2">Usabilidade</td></tr>
<tr><td colspan="2">Gerência de Resolução de Problemas</td></tr>
<tr><td colspan="2">Gerência de Solicitação de Mudanças</td></tr>
<tr><td colspan="2">Avaliação do Produto</td></tr>
<tr><td colspan="4">Processos Organizacionais</td></tr>
<tr><td>Gerência</td><td>Engenharia de Domínio</td><td rowspan="3">Melhoria</td></tr>
<tr><td>Gestão de Ativos</td><td>Infra-estrutura</td></tr>
<tr><td>Gestão de Programa de Reúso</td><td>Recursos Humanos</td></tr>
</table>

Figura 2-5 – Estrutura da Norma ISO/IEC 12207 (Oliveira, 2008).

### 2.5.2 **Norma ISO/IEC 15504 – (SPICE)**

O projeto SPICE (*Software Process Improvement and Capability Determination*) tem as suas origens bem próximas do SPA (*Software Process Assment*). Ambos surgiram do fato das organizações, a nível mundial, estarem fortemente dependentes do uso da tecnologia da informação para automatizar suas operações e seus processos de negócio, e no crescimento da frustação, por parte dos usuários, quando o produto de software é entregue. As falhas em produtos de software vem sendo uma das principais fontes de gastos nas organizações. (NBR, apud Oliveira, 2006).

Outro aspecto que motivou o SPICE foi o surgimento, na década de 80, de iniciativas por parte dos governos dos Estados Unidos e Inglaterra de melhorar o processo de seleção dos seus fornecedores de software. Tal movimento tinha como objetivo conter o crescimento dos custos associados ao software, bem como, reduzir os riscos associados aos projetos de

software e melhorar a qualidade dos produtos finais. Desde o início dos anos 90, um grande número de métodos para melhoria de processos e determinação da capacitação começaram a ser desenvolvidos em vários países. Dentre eles, destacam-se iniciativas como: CMM, TRILLIUM e BOOTSTRAP (Zahran, 1998). Porém, a necessidade da padronização de um modelo internacional fez-se necessário não só por usar um modelo amplamente aceito, mas também pela possibilidade de comparação de resultados.

A ISO/IEC 15504 possui o objetivo de fornecer uma arquitetura para avaliação de processo de software, devendo ser utilizada por organizações envolvidas em planejar, gerenciar, monitorar, controlar e melhorar a aquisição, o fornecimento, o desenvolvimento, a operação, a evolução e o suporte de software (ISO/IEC, 2008).

O primeiro propósito é permitir o entendimento, por uma organização, do estado dos seus processos, visando estabelecer melhorias. O segundo propósito é determinar a adequação dos processos de uma organização para atender a um requisito particular ou classe de requisitos e o terceiro consiste em determinar a adequação de processos da organização a um contrato ou classe de contratos.

### 2.5.3 ***Capability Maturity Model* (CMM)**

Desenvolvido pelo *Software Engineering Institute* (SEI), em resposta a uma necessidade do governo dos Estados Unidos em avaliar a capacitação de seus fornecedores de softwares, o CMM propõe a avaliação e melhoria da capacitação de uma organização. Para isso, descreve uma arquitetura de maturidade de processos em cinco níveis, onde as organizações passariam de uma cultura de processos *adhoc* para uma cultura de processos disciplinados, onde se pratica melhorias contínuas (Paulk, apud Oliveira, 2006).

No CMM cada nível de maturidade indica uma capacitação do processo e, com exceção do primeiro nível, os demais são compostos de várias áreas-chave de processo. No CMM KPA identifica um conjunto de atividades relacionadas que, quando coletivamente executadas, alcançam o conjunto de objetivos considerados importantes para melhorar a capacitação dos processos, onde, cada KPA é organizada em cinco seções denominadas “características comuns” (atributos que indicam se a implementação e institucionalização de uma KPA é efetiva, repetitível e duradoura). Características comuns por sua vez contém práticas-chave que quando executadas em conjunto, alcançam os objetivos das KPAs, como mostrada na Figura 2-6

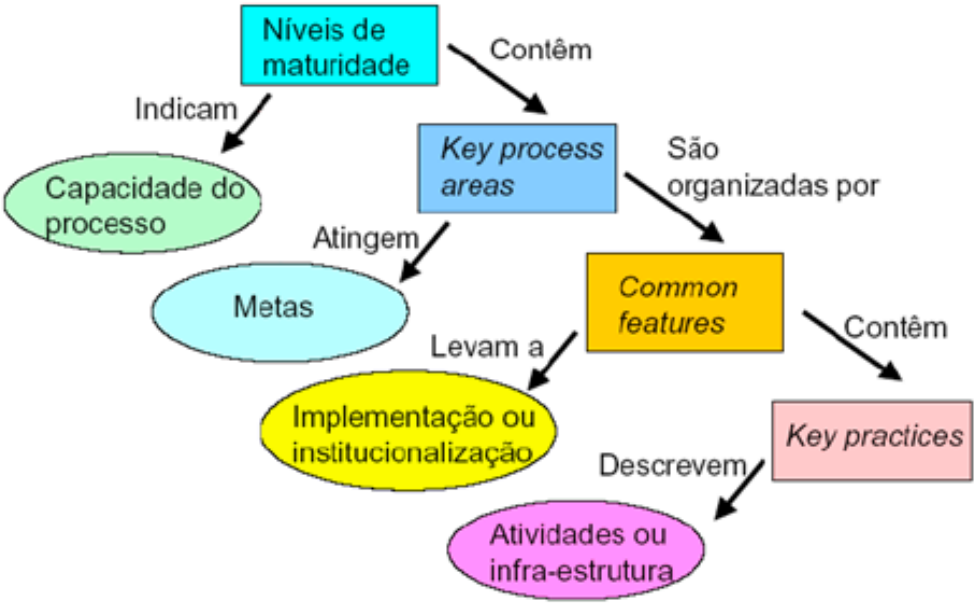


Figura 2-6 – Estrutura do CMM (Adaptada de Paulk, apud Oliveira, 2006).

As KPAs indicam o que deve ser realizado para alcançar um nível de maturidade, onde, uma organização pertencente a um determinado nível (maior que o primeiro nível), deve realizar todas as KPAs deste nível e dos níveis antecedentes.

Abaixo são apresentados os níveis de maturidade do CMM, são eles (Oliveira, 2006):

• **Nível 1 Inicial**: o Processo de Software é caracterizado como *adhoc* e, eventualmente, caótico. Poucos processos são definidos, e o sucesso depende de esforços individuais e heroísmo. Não existem KPAs associadas;

• **Nível 2 Repetitivo**: os processos básicos de gerenciamento são estabelecidos para acompanhar custo, cronograma e funcionalidade. Os sucessos em projetos anteriores com aplicações similares são repetidos. As KPAs do nível 2 objetivam questões relacionadas ao estabelecimento de controles básicos de gerência do projeto como: Gerências de Requisitos, Planejamento do Projeto, Acompanhamento do Projeto, Gerência de Subcontratos, Garantia da Qualidade e Gerência de Configuração;

• **Nível 3 Definido**: o processo de software em relação às atividades de gerenciamento e desenvolvimento é documentado, padronizado e integrado em um processo de software padrão para a organização. Todos os projetos utilizam uma versão aprovada e adaptada do processo padrão para desenvolvimento e manutenção de software. As KPAs do nível 3 são: Foco no Processo da Organização, Definição do Processo da Organização, Programa de Treinamento, Gerenciamento Integrado do software, Engenharia de Produtos de Software, Coordenação entre Grupos e Revisões;

• **Nível 4 Gerenciado**: são coletadas medições detalhadas do processo de software e da qualidade do produto. Tanto o processo de software como os produtos são quantitativamente entendidos e controlados. As seguintes KPAs são previstas: Gerenciamento Quantitativo do Processo e Gerenciamento da Qualidade do Software;

• **Nível 5 Otimizado**: melhorias contínuas nos processos são estabelecidas pelo retorno quantitativo dos processos e conduzidas pela introdução de idéias e tecnologias inovadoras. As seguintes KPAs são previstas: Prevenção de Defeitos, Gerenciamento de Mudanças Tecnológicas e Gerenciamento de Mudanças no Processo;

Todo produto de software produzido deve possuir um padrão de qualidade definido entre o contratante e a contratada. Para a construção são necessárias várias fases e atividades, que por sua vez influenciam na qualidade do produto final. A execução ineficiente ou a não execução de uma ou mais atividades pode gerar um produto deficiente, ou seja, fora do padrão de qualidade acordado. Sendo assim, para garantir a qualidade do produto é importante que o processo seja aplicado corretamente (Pfleeger, 2004).

### 2.5.4 *Capability Maturity Model Integration* (CMMI)

O CMMI possui como propósito fornecer guias para o melhoramento de processos e para o gerenciamento do desenvolvimento, aquisição e manutenção de produtos e serviços (CMMI, 2011). O CMMI consiste em um modelo de avaliação baseado em maturidade, e atualmente aplicável a diversas organizações tecnológicas: quer produzam software ou sistemas, quer executem projetos inovadores ou operações contínuas.

O CMMI possui duas representações: contínua e em estágios. A representação contínua oferece uma abordagem mais flexível para melhoria do processo, onde, são focadas áreas de processo específicas diretamente relacionadas ao objetivo de negócio da organização igualmente a ISO/IEC 15504. Já a representação por estágios oferece um passo a passo detalhado para melhoria do processo, descreve a sequência de execução das áreas de processo, e estas são agrupadas em níveis de maturidade que quando alcançados indicam uma melhoria substancial do processo.

Os componentes da representação em estágios do modelo CMMI são: Níveis de Maturidade, Áreas de Processo, Objetivos Específicos e Genéricos, Práticas Específicas e Genéricas, e Características Comuns.

Na representação em estágios os níveis de maturidade possuem cinco níveis, são eles:

• **Nível 1 Inicial:** os processos são usualmente caóticos e adhoc;

- **Nível 2 Gerenciado**: os requisitos são gerenciados e os processos são planejados, realizados, medidos e controlados;
- **Nível 3 Definido:** os processos são bem caracterizados e entendidos, e são descritos em termos de padrões, procedimentos, ferramentas e métodos;
- **Nível 4 Gerenciado Quantitativamente:** objetivos quantitativos para qualidade e desempenho do projeto são estabelecidos e usados como critério nos processos de gerenciamento;
- **Nível 5 Otimizado**: os processos são continuamente melhorados;

Cada nível de maturidade é definido por um conjunto de áreas de processo onde, uma área de processo é um conjunto de práticas relatadas que, quando cumpridas coletivamente, satisfazem um conjunto de objetivos considerados importantes para se ter uma melhoria significativa naquela área.

### 2.5.5 **Modelo de Referência MPS.BR (Melhoria de Processo de Software Brasileiro)**

O MPS.BR é um programa para Melhoria de Processo do Software Brasileiro coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), contando como o apoio do Ministério da Ciência e Tecnologia (MCT), da financiadora de Estudos e Projetos (FINEP) e do Banco Interamericano de Desenvolvimento (BID) (Softex, 2011).

Em 2003 dados da Secretaria Política de Informática do Ministério da Ciência e Tecnologia mostravam que 30 empresas possuíam certificação CMMI destas apenas 24 possuíam nível 2, cinco estavam classificadas no nível 3, uma no nível 4 e nenhuma no nível 5. O pequeno número de empresas com certificação CMMI da-se principalmente ao custo elevado (Softex, 2011).

Fundamentalmente, o projeto MPB.BR surgiu visando  a criação e disseminação do Modelo de Referência para Melhoria do Processo de Software (MR MPS). O MPS.BR não possui como objetivo definir algo novo no que se refere a Normas e Modelos de Maturidade. A novidade do projeto está na estratégia adotada para sua implementação, criada para a realidade brasileira sendo aderente aos outros padrões de qualidade (CMMI, ISO/IEC 12207, ISO/IEC 15504) aceitos internacionalmente (Weber, 2004).

Umas das metas do MPS.BR é definir e aprimorar um modelo de melhoria e avaliação de processo de software, atendendo as necessidades das empresas. Outra meta do MPS.BR é

ser reconhecido nacional e internacionalmente como um modelo aplicável à indústria de software.

O MPS.BR estabelece um modelo de processos de software e um método de avaliação de processos que dá sustentação e garante que o MPS.BR está sendo empregado de forma coerente com as suas definições. Além, disso o MPS.BR estabelece também um modelo de negócio para apoiar a sua adoção pelas empresas brasileiras desenvolvedoras de software.

Para o desenvolvimento de suas atividades, o MPS.BR conta com duas estruturas, a primeira é o Fórum de Credenciamento e Controle (FCC) e a segunda é a Equipe Técnica do Modelo (ETM). O FCC tem como principal objetivo assegurar que as Instituições Implementadoras (II) e Instituições Avaliadoras (IA) sejam submetidas a processos adequados de credenciamento e que suas atuações não se afastem dos modelos éticos e de qualidades esperados, além de avaliar e atuar sobre o controle dos resultados obtidos pelo MPS.BR (Softex, 2011).

A outra estrutura é a ETM que possui como objetivos atuar sobre os aspectos técnicos relacionados ao Modelo de Referencia (MR-MPS) e Modelo de Avaliação (MA-MPS), tais como a concepção e evolução do modelo, elaboração e atualização dos guias do MPS.BR, preparação de material e definição da forma de treinamento e de aplicação de provas, publicação de relatórios técnicos e interação com a comunidade visando a identificação e aplicação de melhores práticas.

A base técnica para a construção e aprimoramento deste modelo de melhoria e avaliação de processo de software é composta pelas normas NBR ISO/IEC 12207 – Processo de Ciclo de Vida de Software, pelas emendas 1 e 2 da norma internacional ISO/IEC 12207 e pela ISO/IEC 15504 – Avaliação de processo. Uma avaliação MPS.BR é realizada utilizando o processo e método de avaliação MA-MPS descritos no guia de avaliação. Uma avaliação MPS.BR verifica a conformidade de uma organização/unidade organizacional aos processos do MR-MPS. O MR-MPS é definido em consonância com a norma internacional ISO/IEC 12207, adaptando-a às necessidades da comunidade de interesse. O MR-MPS foi definido em conformidade ao CCMI-DEV. Para definição e revisão do modelo de referência é feita uma ampla consulta à comunidade de implementadores e avaliadores MPS.BR.

O MPS está dividido em três componentes são eles:

- Modelo de Referencia (MR.MPS);
- Modelo de Avaliação (MA.MPS);
- Modelo de Negócio (MN.MPS).

O modelo de Referência MR-MPS contém os requisitos que os processos das unidades organizacionais devem atender para estar em conformidade como MR-MPS. Ele contém as definições dos níveis de maturidade, processos e atributos do processo. O MR-MPS está em conformidade com os requisitos de modelos de referência de processo da norma ISO/IEC 15504-2 (Softex, 2011). O MR-MPS é ainda composto por mais três documentos são eles:

- **Guia de Aquisição**: é um documento destinado à organização que pretende adquirir software e serviços correlatos. O guia de Aquisição não contém requisitos do MR-MPS, mas boas práticas para a aquisição de software e serviços correlatos;
- **Guia de Implementação**: fornece orientações para implementar nas organizações os níveis de maturidade descritos no Modelo de Referencia MR-MPS, detalhando os processos contemplados nos respectivos níveis de maturidade e os resultados esperados com a implementação dos processos. O Guia de Implementação está subdividido em 7 partes, contemplando respectivamente, os sete níveis de maturidade(Softex, 2011);
- **Guia de Avaliação**: contém o processo e o método de avaliação MA-MPS, os requisitos para os avaliadores lideres, avaliadores adjuntos e instituições Avaliadoras (IA). O processo e o método de avaliação MA-MPS estão em conformidade com a norma ISO/IEC 15504-2;

O Modelo de Negócio MN-MPS, descreve regras de negócio para implementação do MR-MPS pelas Instituições Implementadoras (II), avaliação seguindo o MA-MPS pelas Instituições Avaliadoras (IA), organização de grupos de empresas para implementação do MR-MPS e avaliação MA-MPS pelas Instituições Organizadoras de Grupos de Empresas (IOGE), certificação de consultores de aquisição e programas anuais de treinamento por meio de cursos, provas e workshops MPS.BR.

Os níveis de maturidade estabelecem patamares de evolução de processos caracterizando estágios de melhoria da implementação de processos na organização. O nível de maturidade de uma organização permite definir seus resultados futuros ao executar um ou mais processos. O MR-MPS define sete níveis de maturidade são eles: A (Em otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado), G (Parcialmente Gerenciado). A escala progride do nível G ao nível A. Para cada nível de maturidade da escala é atribuído um perfil de processo que indicam onde a organização deve concentrar esforços de melhoria. Para que a organização alcance um determinado nível de maturidade, a mesma deve alcançar os propósitos e os

resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível.

Os estágios do MPS-BR são aderentes aos dos CMMI-DEV porém possuem uma graduação diferente, com o objetivo de facilitar a implementação e possibilitar uma avaliação mais adequada a micro, pequena e médias empresas. O fato do MPS.BR possuir mais níveis de maturidade do que o CMMI, proporciona as empresas realizarem avaliações em um período menor, desta maneira, dando uma maior visibilidade dos resultados de melhoria de processos.

## 2.6 Qualidade do Processo x Qualidade do Produto

Em um processo de software são muitas as atividades que afetam diretamente a qualidade do produto. A execução ineficiente ou a não execução de algumas delas pode acarretar a obtenção de um produto de software que não esteja alinhado com as expectativas e requisitos dos usuários. Com isso, é amplamente aceito que a qualidade do produto seja fortemente dependendo da qualidade da execução do processo que o gerou (Pfleeger apud Oliveira, 2006).

Características de qualidade de um produto de software são os atributos nos quais sua qualidade pode ser descrita e avaliada. A Norma ISO/IEC 9126, (2003) objetiva fornecer uma arquitetura para avaliação da qualidade do produto de software. Para atender tal objetivo, a Norma define um modelo de qualidade aplicável a qualquer tipo de software. São definidas seis características de qualidade, são elas:

- Funcionalidade: trata a existência de um conjunto de funções que satisfazem necessidades declaradas ou implícitas, e suas propriedades específicas;
- Confiabilidade: trata da capacidade do software de manter seu nível de desempenho sob condições estabelecidas por um Período definido de tempo;
- Usabilidade: trata do esforço necessário para usar o software, e executar avaliações individuais do referido uso, por um conjunto de usuários definidos;
- Eficiência: trata da relação entre nível de desempenho do software e a quantidade de recursos utilizados, sob condições estabelecidas;
- Manutenabilidade: trata do esforço necessário para a realização de modificações específicas no software;
- Portabilidade: trata da habilidade do software em ser transferido de um ambiente para o outro.

Qualidade de Software não se atinge de forma espontânea, a qualidade de Software depende fortemente da qualidade do processo de desenvolvimento de software usados para desenvolve-los. Um bom Processo de Software não garante que os produtos de software produzidos pela organização são de boa qualidade, mas é um indicativo que a organização é capaz de produzir bons produtos de software (Oliveira, 2006).

## 2.7 Considerações finais

A qualidade do Processo de desenvolvimento de software influi diretamente na qualidade do produto produzido por este processo, pelo fato de um software possuir característica de um produto de difícil avaliação de qualidade e difícil prevenção de resultados, torna-se de extrema importância certificar a qualidade do processo de fabricação deste software (produto). Uma vez tendo boas práticas e atividades estudadas para a produção deste produto torna-se mais provável que o mesmo tenha como resultados os esperados os pretendidos pelos desenvolvedores e esperado pelos clientes.

# Capítulo 3 - **Teste de Software: Uma Visão Geral**

A partir da década de 90, e principalmente nos dias atuais, com a utilização da Internet para a realização de negócios, houve uma mudança na abrangência e complexidade das aplicações onde fatores como segurança e desempenho passaram a ser relevantes, tornando a atividade de testar cada vez mais especializada. Do ponto de vista psicológico, teste de software poderia ser visto como destrutivo em vez de construtivo, pois, em um primeiro momento, o engenheiro tenta construir um software a partir de um conceito até obter um produto tangível. Quando então é passado para etapa de testes (Pressman, 2007).

No propósito de melhorar a qualidade e desenvolvimento de seus softwares, as empresas têm voltado suas atenções para os processos de testes, proporcionando uma grande economia em correções de software mal produzidos. Quanto antes detectada a presença de um defeito, menor torna-se o custo de correção e maior a probabilidade de corrigi-lo corretamente.

Este capítulo tem como objetivo dissertar mas não exaurir a atividade de Teste de Software onde está dividido em 6 seções. A seção de considerações iniciais que tratará dos conceitos relacionados a testes bem como sua importância dentro das organizações e satisfação dos clientes. A segunda seção onde será apresentada as técnicas de teste bem como os tipos de testes que podem ser aplicados associados a essas técnicas. A seção três, que discutirá os processos de teste de software que podem ser executados dentro de uma organização. Já a quarta seção mostrará os modelos e frameworks mais utilizados e um case de adaptação. E por fim as considerações finais onde será mostrado a conclusão deste capítulo.

## 3.1 A Considerações iniciais

Durante as décadas de 70, 80 e 90, os testes eram efetuados pelos próprios desenvolvedores do software, cobrindo aquilo que hoje chamamos de testes unitários e testes de integração. As informações extraídas não permitiam que todos os defeitos fossem encontrados, muitas vezes, os próprios usuários eram envolvidos  para aprovar os resultados desses testes ou mesmo participar da criação dos dados de testes. Essa prática gerava uma

incidência de defeitos alta, que apareciam quando o software já estava em produção o que tornava seu custo de correção bastante elevado (Bastos, 2007).

Ainda segundo Bastos, (2007), os problemas atingiram patamares ainda maiores quando começaram a surgir os sistemas voltados a internet e a imagem das empresas passaram a ser cada vez mais expostas ao grande público. Além disso, essas aplicações se tornaram ainda mais complexas, devido ao surgimento de novas tecnologias o que levou as organizações a procurar novos caminhos para melhorar a qualidade do software desenvolvido. Um desses caminhos foi o aprimoramento da atividade de teste, onde trouxe a necessidade da criação de um processo paralelo, contando com pessoal especializado, que pudesse suportar os projetos de testes, que, por sua vez estavam vinculados aos projetos de desenvolvimento. Os resultados desta prática foram imediatos quanto a qualidade do produto entregue. Portanto, cada vez mais empresas passaram a adotar esse modelo, que significa, num curto prazo, também a redução dos custos de manutenção dos software.

Segundo Hetzel, (1987), o teste de software é uma das atividades de verificação e validação cujo objetivo é avaliar uma característica ou recurso de um programa ou sistema, sendo considerada elemento crítico para a Garantia de Qualidade do Software. Para Myers, (1979), teste é o processo de execução de um programa com a intenção de achar defeitos. E a intenção estaria associada a adição de algum valor ao programa, ou seja, melhorar a qualidade ou a segurança do programa.

A importância de se testar software vem de um passado onde, a ocorrência de BUG's [3]levaram vários planos ao fracasso. Em 3 de novembro de 1999 a sonda da NASA *Mars Polar Lander* desaparecia em pouso em Marte em função de um bit não esperado no conjunto de instruções do programa.

Outro exemplo, foi o míssil de guerra *patriot*, usado como parte do sistema estratégico de defesa dos EUA, do governo de *Ronald Reagan*, que teve problemas sérios quando foi usado efetivamente na defesa do míssil *Scud* do Iraque acertando vários alvos errados, em função de problemas no temporizador do sistema de horas do míssil, onde, quando eram acumuladas mais de 14 horas, acarretava um erro no sistema de rastreamento.

---

[3] Bug é um erro no funcionamento comum de um software, também chamado de falha na lógica programacional de um programa de computador, e pode causar discrepâncias no objetivo, ou impossibilidade de realização, de uma ação na utilização de um programa de computador (Molinari, 2006).

Os exemplos citados acima servem para mostrar que um teste bem executado diminui as chances de ocorrer BUG's, diminuindo também as chances de falhas do negócio. Segundo Molinari (2006), BUG ou Falha, é um problema ou qualquer falha no software. Por exemplo, um jogo que não funciona em um computador. Já no conceito de Defeito, Molinari direciona que um defeito é qualquer não conformidade em relação ao que o software se propõe a fazer.

## 3.2 Técnicas de Teste de Software

As técnicas de teste são geralmente divididas em duas: Técnicas de Teste Estrutural e Técnicas de Testes Funcional; cada uma com objetivos distintos e tipos de testes distintos. A técnica de software estrutural consiste em garantir que os softwares e os programas sejam estruturalmente sólidos e que funcionem no contexto técnico onde serão instalados. Já a técnica de teste funcional garante o atendimento aos requisitos, ou seja, que os requisitos estão corretamente codificados (Rios, 2006). Nesta seção serão apresentadas as duas técnicas de teste e os tipos de testes participantes de cada uma.

### 3.2.1 **Técnicas de Teste de Software Estrutural (Caixa Branca)**

O objetivo dos testes estruturais é garantir que o produto seja estruturalmente sólido e que funcione corretamente, busca-se assim, determinar se a tecnologia foi usada de modo adequado e seus componentes montados funcionam de forma coesa (Rios, 2006). Ainda segundo Molinari, (2006), esta técnica possui também como objetivo garantir que todas as linhas de códigos, condições de decisão e laços sejam executadas pelo menos uma vez e estejam corretas.

Existem vários tipos de testes que utilizam a técnica de Caixa Branca, abaixo são citados e explicados três tipos:

- **Teste de Unidade**: também conhecido como teste unitário ou teste de módulo, se testa as menores unidades de software desenvolvidas (Classes). O universo alvo deste tipo de teste são as subrotinas ou mesmo pequenos trechos de código. Assim, o objetivo é de encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independente do todo;
- **Teste de Integração**: possui como objetivo encontrar falhas provenientes da integração interna dos componentes de um sistema. Geralmente os tipos de falhas encontradas são de transmissão de dados. Por exemplo, um componente A aguardando um retorno de um

valor X ao executar um método do componente B; porém, B pode retornar um valor Y, gerando um falha;

- **Inspeção de código**: é um tipo de teste que possui objetivo de identificar possíveis desvios de padrões de codificação estabelecido. Um exemplo, é inspecionar o código quanto ao padrão de nomes de classes;

### 3.2.2 **Técnicas de Teste de Software Funcional (Caixa Preta)**

Os testes funcionais do sistema são desenhados para garantir que os requisitos e as especificações do sistema tenham sido atendidos. O processo costuma envolver as criações das condições de teste para uso na avaliação da correção da aplicação (Rios, 2006). Também chamada de teste funcional, orientado a dado ou orientado a entrada ou saída, a técnica de caixa preta avalia o comportamento externo do componente de software, sem considerar o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Vários tipos de testes funcionais são conhecidos. Abaixo são apresentados os mais utilizados:

- **Teste de Funcionalidade:** possui como objetivo testar as funcionalidades na visão do usuário através de utilização do sistema guiado por Casos de Testes planejados e pré estabelecidos;
- **Teste de Sistema:** possui como objetivo executar o sistema sob ponto de vista de seu usuário final, executando as funcionalidades em busca de falhas em relação aos seu objetivos originais;
- **Teste de Conformidade:** possui como objetivo testar, na visão do usuário, se o sistema apresenta o comportamento especificado; Considerações Finais.

## 3.3 Técnicas de Teste de Software

O teste, da maneira como é executado pela maioria das empresas (como etapa dentro do processo de desenvolvimento e, em geral, executado pelos próprios desenvolvedores e pelos usuários do sistema), serve apenas para garantir que as especificações ou os requisitos do negócio foram de fato implementados. No modelo de qualidade apenas essa prática é insuficiente, não dando para garantir que o software testado apenas pelos desenvolvedores

está corretamente testado. Portanto a melhor maneira de se testar um software é ter um processo de teste de software claramente definido (Rios, 2006).

Geralmente nos processos de desenvolvimento de aplicações, os prazos são apertados e, com frequência previamente estabelecidos. Nem sempre há prazos específicos para os testes, isto é, os testes precisam ser feitos dentro do prazo estipulado para o desenvolvimento. Em uma situação crítica como essa, é muito importante o uso de uma metodologia adequada e de uma equipe treinada e capacitada. Desse modo, através de revisão e inspeção, os testes podem ser feitos nos documentos de desenvolvimento tão logo o projeto de desenvolvimento de software se inicia.

A importância da descoberta prévia de defeitos é justificada pela Regra 10 de Myers, que estabelece que o custo de correção de defeitos tende a aumentar quanto mais tarde (produção) o defeito é detectado como mostra a Figura 3-1.

Bem como outras atividades, o teste de software possui processo e modelos próprios, onde podem ser instanciados de acordo com a necessidade de cada organização ou cada processo de desenvolvimento.

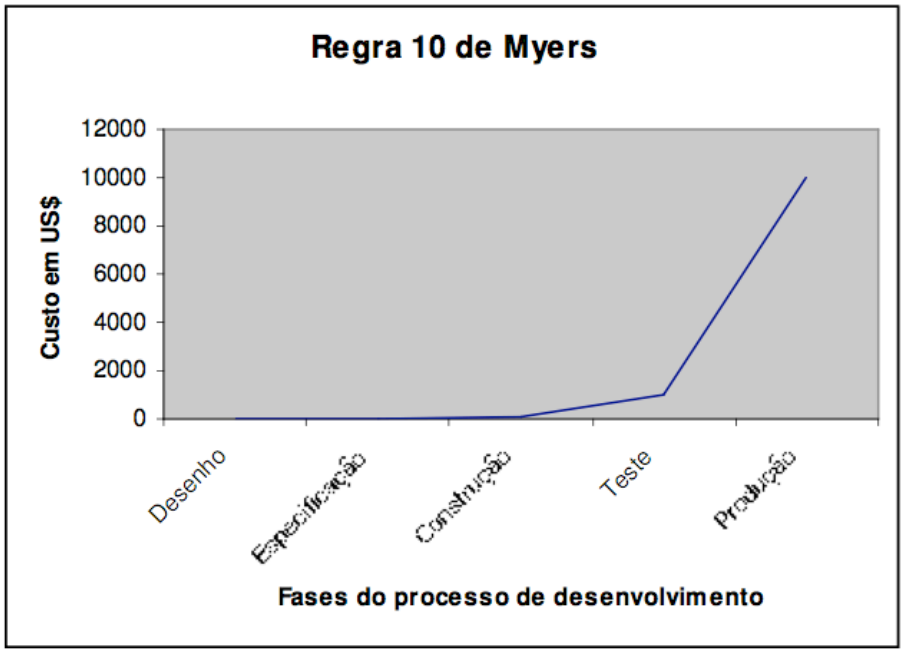


Figura 3-1 – Regra 10 de Myers(Rios, 2006).

### 3.3.1 **Modelo *Waterfall***

Um dos primeiros modelos de desenvolvimento de software foi o *Waterfall* (ou modelo em cascata, como traduzido pela maioria dos autores), como mostrado na Figura 3-2.

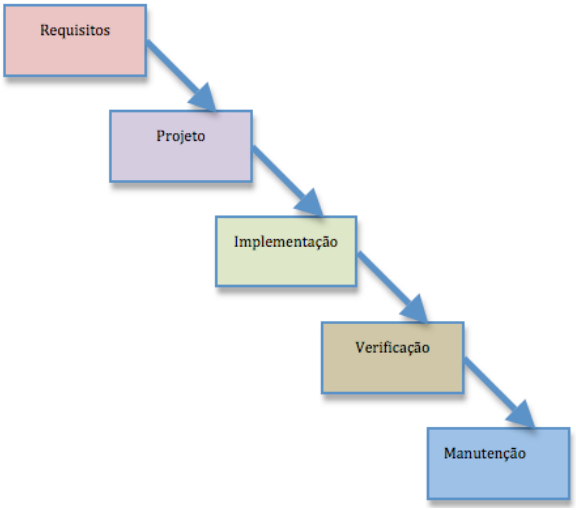


Figura 3-2 – Modelo de Teste *Waterfall* (adaptado de Molinari, 2006).

As fases individuais ou atividades, segundo, Molinari, (2006), foram definidas de modo a representar a linha de desenvolvimento vigente na época. Cada conjunto de atividades deveria ser completado como um todo antes de passar para o próximo conjunto de atividades. O retorno a uma fase qualquer implicava em passar por todas as todas as outras anteriores.

No caso em questão, as atividades de testes somente começariam após a fase de implementação. A grande desvantagem para a qualidade do produto é que os testes eram a última fase antes do release do produto. Com isso os testes se tornavam de certa forma uma fase pela qual "todos queriam passar rapidamente", ou ainda quando havia algum atraso em alguma das atividades anteriores, geralmente a atividade que era penalizada era a de testes por ser a última, desta maneira, acarretando um encurtamento da atividade de teste, e por conseqüência perda de qualidade do produto.

### 3.3.2 **Modelo *SawTooth***

Segundo Molinari, (2006), esse modelo, visto na Figura 3-3 mostra uma integração entre o usuário e o desenvolvedor. O desenvolvimento tem aqui um enfoque simples, semelhante ao de um pedreiro que faz uma obra numa casa: faz por encomenda, isto é, faz o que foi pedido. Como o desenvolvedor está sobrecarregado de tarefas que não seriam somente dele, acaba que vários pontos de qualidade do projeto ficam a desejar. Pode-se chamar esse modelo de "Serra" ou "Visão Dentária", pois faz lembrar a parte inferior de uma arcada dentária cujos dentes são pontiagudos. A fase de testes é compreendida após o termino da implementação, onde, então são executados os testes de caixa preta.

Mesmo os desenvolvedores estando ocupado com tarefas que não seriam somente dele, o fato de haver a apresentação de protótipos para o usuário reduz, o risco de o software sair fora do escopo requerido pelo usuário.

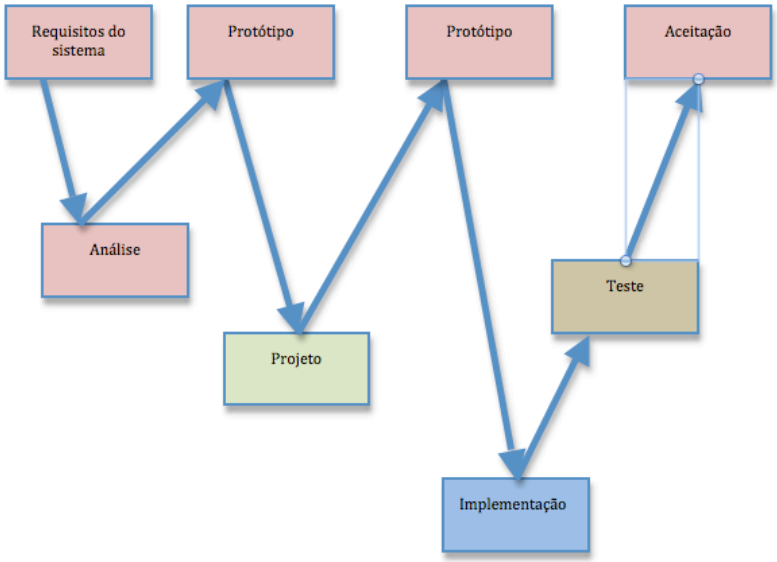


Figura 3-3 – Modelo de Teste de Sotware *SawTooth*. (adaptado de Molinari, 2006).

### 3.3.3 **Modelo *SharkTooth***

Esse modelo é semelhante ao *SawTooth Model*, porém possui agora uma participação de um "gerente" que entra como um revisor no processo, de modo a tentar minimizar o "gap" de padrões de desenvolvimento e qualidade do produto junto ao desenvolvimento. Dependendo do caso, passa a ser mais um complicador (Molinari, 2006).

A diferença entre esse modelo e o *SawTooth Model*, é que quem fará a integração do sistema e seu respectivo teste é o gerente, "Alguém mais experiente". Esse modelo pode também ser chamado de "boca de tubarão" ou "testes de tubarão" por parecer com um "sorriso de um tubarão". O modelo apresenta muita característica política, pois o gerente participa ativamente em nível operacional para mostrar coisas do tipo "Sr. Usuário o sistema vai sair do jeito que foi pedido, estou participando ativamente do processo".

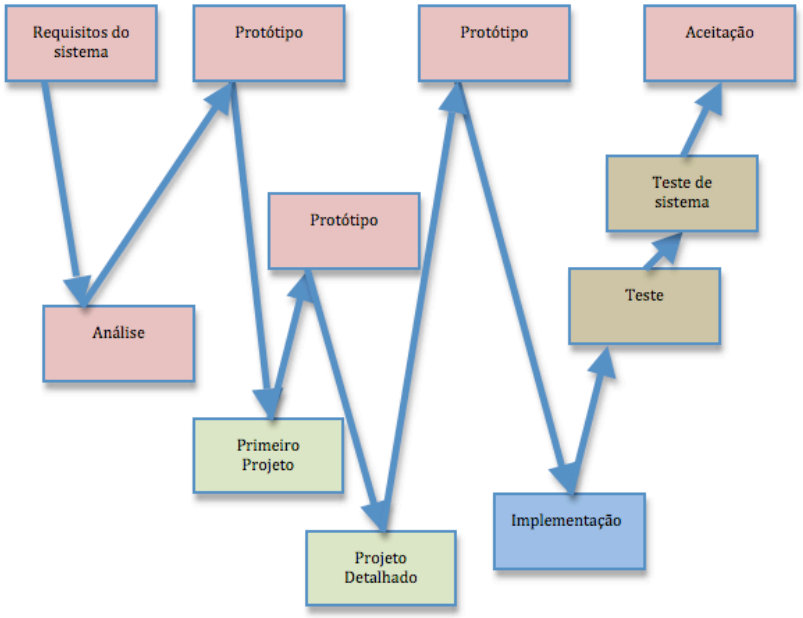


Figura 3-4 – Modelo de Teste de Sotware *SharkTooth*. (adaptado de Molinari, 2006).

### 3.3.4 **Modelo *Espiral***

O modelo em espiral na verdade é cíclico e faz uso da prototipação. Nesse modelo, os testes estão devidamente explicados (análise de risco, validação de requerimentos e desenvolvimento, testes) e está dividido em estágios: modular, integração e testes de aceitação (Molinari, 2006). O detalhe é que nesse modelo os testes seguem a linha de codificação. A execução é que o plano de teste deve ser construído depois do projeto do sistema. O modelo em espiral não identifica atividades associadas com remoção de defeitos, Figura 3-5.

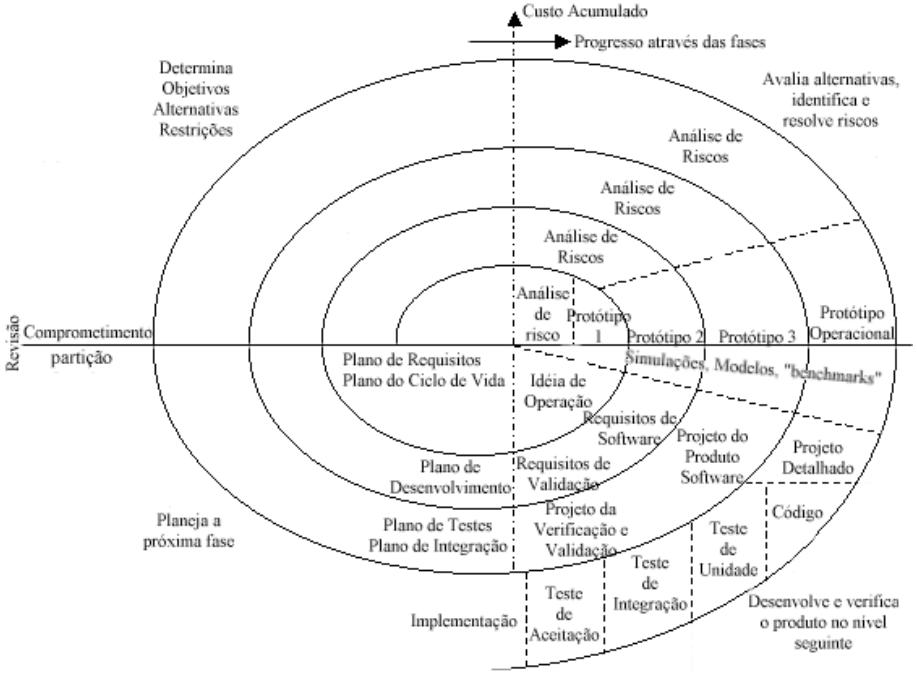

Figura 3-5 –Modelo de Teste de Software Espiral (adaptado de Molinari, (2006)).

### 3.3.5 **Modelo "V"**

Segundo Molinari, (2006), pode-se dizer que a maioria dos modelos de processo conecta-se "graficamente" a esse modelo, que se tornou um dos mais populares. Ele descreve "graficamente" as fases individualmente em forma de "V". Nesse caso "V" vem de verificação e validação, Figura 3-6.

Esse modelo é simples e fácil de ser entendido. As atividades são ordenadas da forma sequencial em níveis de abstrações de modo que as conexões com as fases de desenvolvimento ficam extremamente claras.

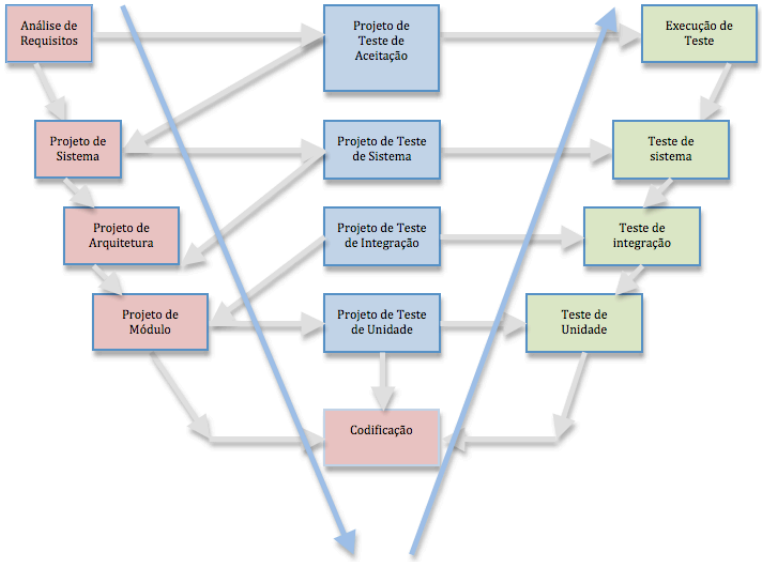


Figura 3-6 –Modelo de Teste de Software "V" (adaptado de Molinari, (2006)).

A desvantagem principal é que do lado esquerdo tem-se "fases" mais "construtivas" enquanto no lado direito tem-se fases mais "destrutivas". Isso faz com que exista uma "concorrência" entre equipe de desenvolvimento e testes. Quem é mais rápido? Aquele que constrói ou aquele que destrói?

Outra desvantagem é que não há um planejamento de remoção de defeitos e testes dos defeitos encontrados. Os testes de regressão não são aplicados em lugar nenhum neste modelo. Não esquecendo que este modelo se tornou o "fusquinha" dos modelos de teste, porque sua simplicidade mostra que se deve fazer em linhas gerais.

### 3.3.6 **Modelo "W"**

Hoje pode-se dizer que esse modelo baseia-se no tão conhecido modelo "V". A diferença é que para cada etapa do modelo "V" tem uma etapa de "testes" que corresponde a cada fase, de maneira que é possível eliminar os "BUG's" desde o início. No lado esquerdo tem-se um planejamento de cada fase, no lado direito tem um teste de cada fase (Molinari, 2006).

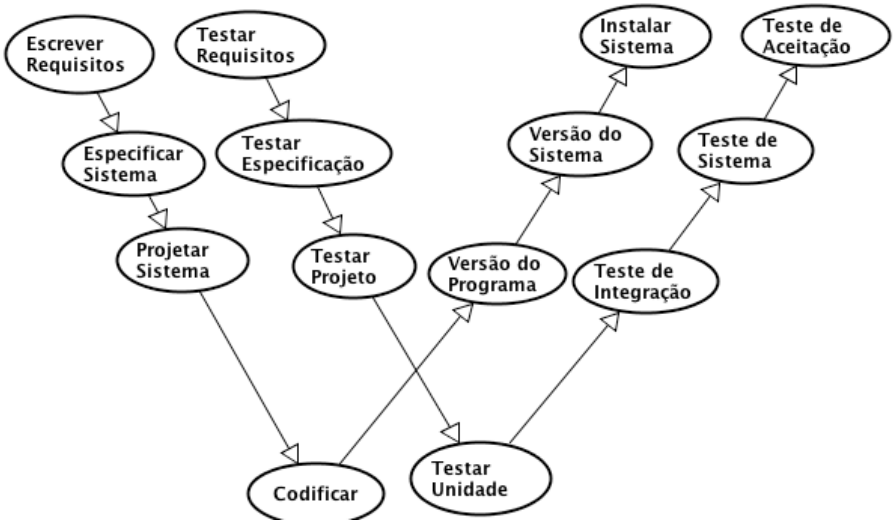


Figura 3-7 –Modelo de Teste de Software "W" (adaptado de Molinari, (2006)).

O fato é que esse modelo visa de forma radical diminuir custos, partindo do princípio que quanto mais se testasse, mais cedo se encontrariam os problemas. Os problemas ficam minimizados (mas não resolvidos), tais como: cada lado possui uma parte construtiva e outra destrutiva. A vantagem do modelo em "W" está onde cada atividade de teste torna-se mais clara.

### 3.3.7 **Modelo *Buterfly***

Esse modelo que também se baseia no modelo "V", parte do pressuposto de que o modelo "V" tenta linearizar uma tarefa não-linear: o desenvolvimento de software. Um exemplo típico é a fase de projeto ("design") no modelo "V" na qual pode haver "macro-feedbacks" que foram simplificados, mas na essência são extremamente importantes. Cada subatividade possui na prática pequenas fases de análise, projeto, execução (todas relativas a testes) (Molinari, 2006).

Figura 3-8 –Modelo *Butefly* de Teste (adaptado de Molinari, (2006)).

Na visão do modelo "Butterfly", cada subfase representaria um componente de um único ser que é mutável  e tem vida curta. Neste caso, uma borboleta. A asa esquerda seria a etapa de análise, a da direita seria análise dos resultados da execução (*feedback)* enquanto que parte central dela seria a execução dos testes propriamente ditos.

Fazendo uma analogia ao modelo V de teste de Software o modelo borboleta seria representado pela Figura 3-9

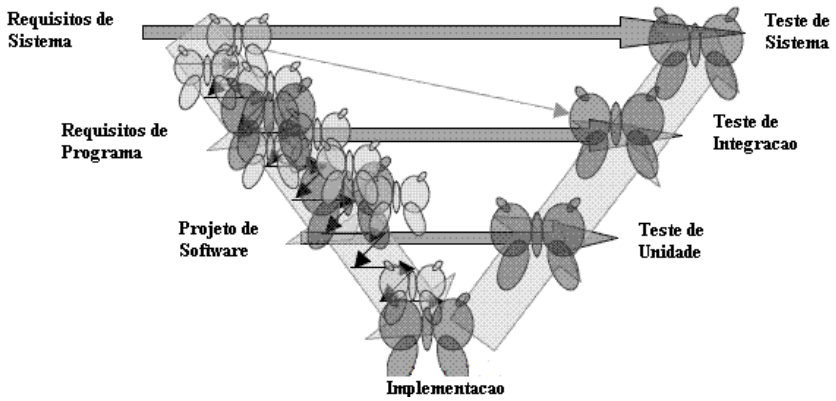


Figura 3-9 –Representação do Modelo *Buterfly* (Molinari, 2006).

### 3.3.8 **Modelo "Y"**

O modelo Y ou "*why-model*" ("Modelo-porque"), usa duas das quatro dimensões de testes e coloca uma terceira dimensão representativa do processo de desenvolvimento. Essas dimensões podem ser a "meta" e o "momento" (Molinari, 2006).

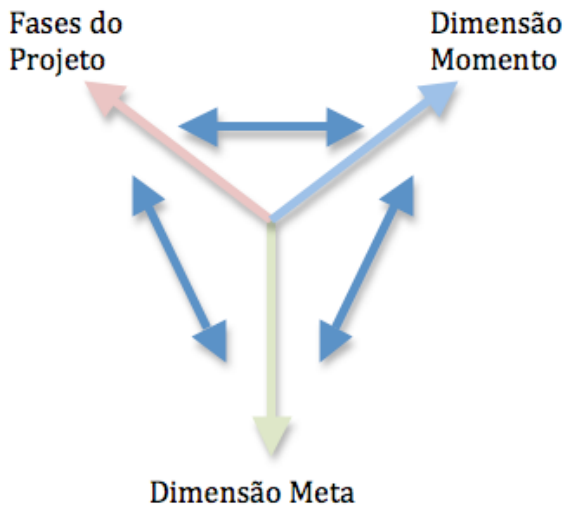


Figura 3-10 –Representação do Modelo “Y” de Teste (Molinari, 2006).

Seria como colocar uma dimensão a mais no modelo “V”. A diferença é que internamente cada subfase assumiria uma atitude semelhante ao modelo “butterfly” (análise, projeto e execução). O modelo “Y” adquire uma quebra de paradigmas de maneira que se o seu modelo não estiver com um desempenho aceitável no seu ambiente, basta alterar uma das dimensões: meta ou momento, o modelo Y é representado na Figura 3-10.

## 3.4 Modelos de Maturidade de Teste de Software

Devido a uma alta demanda de insatisfações na qualidade de software, as organizações têm a difícil tarefa de melhorar o processo de desenvolvimento de software, entregando-os com qualidade, dentro de prazos e custos controlados e previamente estabelecidos. Na medida em que o emprego de sistemas de informação pela sociedade cresce ao ponto em que boa parte dos negócios depende cada vez mais de software e computadores, passa a ser de vital importância contar com software de qualidade, que fornece resultado correto quando alimentado com dados válidos e que identifica corretamente dados de entrada inválidos. A área de teste de software é fundamental para avaliação da qualidade do software desenvolvido (Ericson, 2009).

A confiabilidade do software é a probabilidade de um programa operar corretamente por um tempo específico em um ambiente específico (Rios, 2006). O custo total dos defeitos que permanecem no software liberado e que eventualmente se manifestam como falhas, têm sido observados com maior criticidade, por muitas organizações, a fim de se fazer uma

estimativa para alocar melhor o esforço ligado aos testes e também reduzir o custo esperado das falhas.

Para melhorar o nível de qualidade e o custo/benefício do teste de software, surgiram os modelos de melhorias de teste de software como: TIM (*Test Improvment Model*), TPI (*Test Process Improvment*), TMM (*Test Maturity Model*), sendo esses os principais modelos e o modelo nacional MPT.BR.

### 3.4.1 **TIM (*Test Improvment Model*)**

O TIM possui como principal objetivo a identificação do estado atual do processo em questão para que possam ser listados e eliminados os pontos fracos. O TIM é dividido em 4 níveis, são eles (Ericson, 2009):

- ***Baselining***: onde consiste na padronização da documentação, métodos e políticas, além de também adotar padrão para a análise e classificação dos problemas;
- ***Cost-efectiveness***: onde adota como melhoria a adoção de atividades que: realizem a detecção de defeitos desde o início do projeto, façam a automação de tarefas de teste, de treinamento e por fim adotem o reuso;
- ***Risk-lowering***: recomenda o envolvimento desde o início do projeto, realiza uma análise de custo/benefícios para ajustar os gastos, realiza também uma análise nos produtos e processos, realiza uma análise e gerenciamento de riscos e a comunicação com todas as etapas e áreas envolvidas no projeto;
- ***Optimizing***: melhoria contínua, análise das causas raízes para os principais problemas, cooperação com todas as partes envolvidas do projeto em todas as fases do desenvolvimento, conhecimento e entendimento através de experimentações e modelagem.

Para a avaliação da aderência do processo de teste com o modelo TIM, é realizada uma entrevista com pessoas chaves participantes do processo de teste de software, com questões para resposta de “sim” e “não” e discussões sobre ausência de respostas. Após, é realizado uma análise da maturidade, identificação de melhorias, para então realizar uma apresentação da solução proposta (Ericson, 2009).

O modelo TIM é dividido em cinco áreas chaves (KA), ou seja, principais áreas alvo da atuação do modelo de maturidade, onde cada área possui atividades divididas nos níveis de maturidade (níveis de medida utilizados para caracterizar a capacidade de um processo), são elas: Organizacional, Planejamento e Monitoramento, *TestWare*, Casos de Teste e Revisão.

#### 3.4.1.1 Área Organizacional

Para área Organizacional, nível zero, antes do nível ***Baseline***, não há diferença entre o desenvolvimento e o teste. Para uma organização estar no nível de ***Baseline*** é preciso que a organização tenha uma equipe própria, com um papel específico para teste e que a equipe de teste seja liderada por um Líder de Teste (Ericson, 2009).

Já para a área organizacional no nível de ***cost-effectiveness***, torna-se necessário que a equipe de testes suba um nível em relação ao nível de ***baseline***. A equipe de teste deve ser independente do desenvolvimento e ainda a equipe deve possuir um tempo para estabilizar determinado módulo entregue e saber exatamente o que seu papel de teste faz. A independência da equipe serve para que a mesma não sofra pressão do Gerente do Projeto para que a entrega ocorra logo. Ainda sim, para detecção de erro, mais precoces, é recomendado que a equipe de testes esteja no mesmo local da equipe de desenvolvimento para que a comunicação entre as equipes seja facilitada. É também recomendado que a equipe de teste direcione-se apenas para um projeto, e de preferência o mesmo projeto da equipe de desenvolvimento que se encontra no mesmo espaço (Eriscon, 2009).

A área organizacional diz que para atingir o nível de ***Risk-lowering*** é necessário que a equipe de testes trabalhe de forma cooperada com a equipe de desenvolvimento para que a equipe de testes compreenda maiores detalhes sobre o desenvolvimento, para que este conhecimento possa ser usado para detecção de locais onde haja maiores potencialidades de erros. O desenvolvimento por sua vez também sai ganhando, uma vez, que ao encontrar estes erros a equipe melhora seu processo de desenvolvimento para que os mesmos não mais voltem a acontecer.

O nível de ***Optimizing***, para a área organizacional está preocupado com a qualidade da atividade de teste. Mesmo a qualidade sendo uma medida subjetiva, a mesma é alcançada pela satisfação das partes participantes do projeto. Uma boa prática para assegurar que a qualidade é alcançada, é que todas a atividades do processo participem de todas as fases do projeto através de equipes multidisciplinares.

#### 3.4.1.2 Área de Planejamento e Monitoramento

A área de Planejamento é uma área de extrema importância para o teste de software, quando os objetivos e as estratégias não são claras, recursos são disperdiçados. Já o monitoramento torna-se importante para o acompanhamento das atividades que são realizadas.

Para o nível de ***baselining*** a atividade de planejamento pede que seja realizado o planejamento básico da tarefa de verificação. A atividade de planejamento deve ser realizada de acordo com a política de documentação da empresa, ou seja, uma abordagem estruturada é necessário. A abordagem estruturada, trás para o planejamento redução de esforço:

- Conhecimento mais amplo sobre o produto que está sendo testado: Para o melhor desenvolvimento do projeto de teste enquanto maior o conhecimento do objeto que está sendo testado melhor o projeto de teste;
- Inconsistências nos requisitos: muitos problemas podem ser descobertos no próprio planejamento dos testes;
- Maior comunicação com as partes envolvidas do projeto.

No nível de ***cost-effectiveness***, segundo Eriscon, (2009), o planejamento dos testes é realizado por um testador especializado na área a fim de garantir que o planejamento realizado é realmente o planejamento que é necessário. O plano realizado pelo testador trata-se de um plano geral onde os planos específicos são realizados com base no plano geral. Os planos específicos são realizados sob demanda e de forma evolutiva. A forma evolutiva tráz os seguintes benefícios:

- Aprender antes de pensar: o testador adquire novos conhecimentos sobre o produto, facilitando o entendimento para um teste mais completo e aprimorado. Enquanto que a abordagem tradicional exige que um plano de teste seja feito no início, o que acarreta na perda de conhecimento pelo testador. Já na abordagem evolutiva o plano de teste só é finalizado após o aprendizado dos testadores;
- Consistência com o requisito: a maior parte dos testes é baseada nos requisitos e durante projetos longos (anos) os requisitos mudam enumeras vezes. A abordagem evolutiva provê um retrabalho menor nos projetos de teste, desde que os mesmos sejam escritos de maneira evolutiva;
- Planejamento de esforço: um plano de teste só deve ser escrito diante de uma real necessidade de maneira a garantir que nenhum plano seja escrito sem que seja realmente necessário. Com isso, o consumo de recursos é menor.

Na fase de ***risk-lowering***, o planejamento dos testes é realizado por um planejador ou um testador experiente, de maneira a diminuir os riscos, e aproveitar experiências para gerenciar dependências, riscos, etc. Para a redução do risco do projeto, uma análise de risco é realizada. Os testes devem ser formalmente revistos e aprovados por todas as fases afetadas.

A organização no nível de otimização, realiza levantamento de custos, recursos e riscos. Métricas são geradas e utilizadas para o planejamento e monitoramento das atividades onde, essas informações podem ser utilizadas, por exemplo, para medir a produtividade dos testes. Outro exemplo de utilização de métricas para medir a efetividade da equipe de testes é o número de erros detectados.

#### 3.4.1.3 Casos de Testes

Todos os testes dinâmicos podem obter ganho de tempo com testes de casos especificados para serem rodados prioritariamente. Desenvolver e selecionar casos de teste é um aspecto de extrema importância para o teste de software, pode-se dizer que a qualidade dos casos de teste determina a qualidade do próprio teste como um todo.

No nível de ***baselining*** a função de testar tem que iniciar um projeto e a documentação de Casos de Teste de acordo com padrões. Os Casos de Testes têm que ser desenvolvidos baseado nos Casos de Uso [4]que vem indicar os objetos de teste. Os Casos de uso são a fundação de todo o projeto e devem ser preenchidos até o fim do projeto. Os requisitos possuem a tendência de mudar o tempo todo especialmente em projetos de longa duração, portanto é importante rotinas de estabilização validando os Casos de Teste quanto às mudanças dos requisitos.

No nível de ***cost-effective***, a função do teste é estruturar os Casos de Teste respeitando os níveis de teste, requisitos e objetivos de teste, de maneira a facilitar o reuso. O reuso de casos de teste, ocorre quando pode-se utilizar o mesmo caso de teste para diferentes níveis de teste, de preferência, segundo o TIM, os casos de teste devem ser documentados com aderência ao padrão IEEE – *Standart dor Software Test Documentation* (IEEE 829, 1998). A eficiência dos teses no nível de ***cost-effective*** dar-se usando técnicas de documentação.

No nível de ***risk-lowering***, os Casos de Teste são ranqueados e selecionados de acordo com a criticidade quanto aos riscos. O ranking dos Casos de Testes influenciam diretamente a execução de testes fazendo com que os riscos sejam os menores possíveis quando os testes forem terminados.

---

[4] Na Engenharia de Software, um caso de uso (ou *use case*) é um tipo de classificador representando uma unidade funcional coerente provida pelo sistema, subsistema, ou classe manifestada por seqüências de mensagens intercambiáveis entre os sistemas e um ou mais atores (Pressman, 2007).

Para o nível de ***optmining*** é feita uma medição para garantir a qualidade do processo de Testes. As métricas são usadas para prover a qualidade do processo.

#### 3.4.1.4 *Testware*

***Testeware***, é o procedimento de teste executado, o suporte de teste, os dados que são usados na execução de testes e a documentação relacionada. A área chave ***Testware*** inclui, também, o gerenciamento de configuração através de ferramentas. As ferramentas dão assistência na desempenho de tarefas repetitivas. A gerência de configuração é importante para um projeto como um todo e não só para uma área de teste.

No nível de ***baselining***, para o ***testware***, a documentação deve ser mantida em um sistema de arquivo seguro. Os resultados obtidos dos testes devem ser disponibilizados para a organização via intranet.

Para o nível de ***cost-effectiveness***, é disponibilizado um repositório de dados controlado por um sistema de banco de dados, onde, para gerência de configuração vem a facilitar o gerenciamento de ativos, desta maneira, contribuindo com a reutilização de *testware* (Ericson, 2009). Com a adoção do Repositório de dados gerenciado por um Banco de dados, torna mais fácil a gerência de configuração do que com um sistema de arquivos.

O nível de ***risk-lowering*** efetivamente aproveita os testes funcionais dentro da gerência de configuração gerando bastante reutilização. Neste nível, ferramentas de suporte de teste regressivo (selecionar e executar casos de testes) são utilizadas, o que faz com que os riscos em função de um teste de regressão automatizados sejam reduzidos. Aprende-se neste nível que a adoção de ferramentas destinadas a testes regressivos provem bons ganhos a organização, melhoramento do desempenho e diminuição de custos e de riscos.

Na gerência de configuração no nível de *optimising* as ferramentas de Gerência de configuração e de desenvolvimento são integradas. O nível de *optmising* utiliza os resultados de testes, na que a equipe de teste “aprenda” e trabalhe para adicionar novos casos de testes. Os objetivos do teste são alcançados e os casos de testes são automatizados, gerados com base em requisitos, perfis operacionais e arquitetura.

#### 3.4.1.5 Revisão

Na área de revisões, revisoes técnicas são envolvidas para inspeção de documentação de software. No nível de ***baselining*** técnicas simples de revisões são utilizadas, os requisitos no mínimo são revisados. Se para organização revisão é nova, treinamentos são providos para

o primeiro projeto. Não vale a pena correr o risco de realizar revisões mal feitas, antes investir em treinamento do que posteriormente ter que ***dispender*** recursos para recuperar a confiança do cliente. Para isso são treinados alguns líderes para que esses consigam treinar o restante da equipe, enquanto os gerentes são treinados para o convencimento de que a revisão é importante.

Para obter o nível de ***cost-effectiveness***, todos os funcionários deveriam ter sido treinados em revisão. Os teste funcionais são iniciados pela revisão técnica. Já para o caso de ***risk-lowering***, medidas de processo, revisão, produtos e recursos são levantados e armazenados em um banco de dados e documentos de testes revistos para diminuir possibilidades de erros. O resultado das análises são utilizados para o próprio teste.

Ao nível de ***optmizing***, processos, produtos e dados de recurso são utilizados para avaliar, melhorar e escolher técnicas de revisão onde a base para isso foi estabelecida a relação risco - nível de redução, com a coleta de dados. A função de teste é capaz de escolher o método de revisão direito.

### 3.4.2 **TMM (*Test Maturiry Model*)**

A estrutura do TMM é em parte baseada no CMM, e a divisão em estágios do CMMi. Sendo esse o maior benefício para as organizações que já conhecem o CMMi. O TMM possui cinco níveis de maturidade que refletem o grau de maturidade do processo de teste. Para cada nível de maturidade um número de áreas de processo são definidos. Uma área de processo é um conjunto de atividades relacionadas com o processo de teste e cada uma dessas atividades sendo implantada corretamente contribuirá para o melhoramento do processo.

Os cinco níveis de maturidade e suas áreas de processo relacionadas do TMM são os seguintes:

- **Nível 1 – Inicial**: os testes são caóticos, não existe processo definido, sendo considerado parte de um *debugging* (procura, análise e remoção de causas de falhas no software). O objetivo dos testes é mostrar que o software funciona sem maiores falhas. Áreas do processo: Nenhuma área do processo é identificada nesse nível;
- **Nível 2 – Definição**: ocorre a definição de um processo de teste e a separação clara do *debugging*. No contexto de estruturação do processo de testes, planos de teste são estabelecidos contendo estratégias de teste. Os testes ainda acontecem tardiamente dentro do ciclo de vida do desenvolvimento. Áreas do processo: Políticas e objetivos, planejamentos dos testes, técnicas, métodos e ambiente de testes;

- **Nível 3 – Integração**: os testes são completamente integrados ao ciclo de vida do software, sendo reconhecido em todos os níveis do Modelo V. Planejamento dos testes acontecem no estágio inicial dos projetos, através de um Plano Geral de Testes. A estratégia de testes é determinada através de técnicas de gerenciamento de riscos e baseada em requisitos funcionais e não funcionais do sistema. Programas de treinamento e revisões fazem parte do processo. Áreas de processo: organização dos testes, programa de treinamento, ciclo de vida, e integração, e controle e monitoramento;
- **Nível 4 – Gerenciamento e Medição**: os testes são completamente definidos, bem fundamentados e medidos. Revisões e inspeções são incorporadas ao ciclo de vida do desenvolvimento e considerados partes dos testes. Os produtos de software são avaliados a partir de critérios de qualidade por características de qualidade, como reusabilidade, usabilidade e mantenabilidade. Casos de testes são armazenados e gerenciados em uma base de dados central para reuso e testes de regressão. Áreas de processo, revisões (*Peer reviews*), medição dos testes e avaliação da qualidade dos testes;
- **Nível 5 – Otimização**: resultados são arquivados com o objetivo de melhoramentos em estágios anteriores, a área de testes é completamente definida através de seu processo e capaz de controlar seus custos e sendo efetivos. No nível 5, métodos e técnicas são otimizados e estão em melhoramento contínuo. A prevenção de defeitos e o controle de qualidade são introduzidos em outras áreas do processo. Existência de procedimentos para escolha e avaliação de ferramentas de testes. Testar é um processo com o objetivo de prevenir defeitos. Áreas do processo: prevenção de defeitos, controle de qualidade e otimização do processo de teste.

Os cinco níveis de maturidade mostram uma evolução de um caótico e indefinido processo de testes para um controlado e otimizado processo de teste.

### 3.4.3 **TPI (*Test Process Improvment*)**

Este modelo foi desenvolvido baseado no conhecimento prático e em experiências no desenvolvimento do processo de teste. Para organizar a eficiência dos testes, diferentes níveis de teste são usados, e cada nível endereça um certo grupo de requisitos ou especificações técnicas ou ainda funcionais. O modelo TPI apresenta os seguintes passos para alcançar a melhoria de processo, são eles:

- **Determinar metas e áreas de consideração:** qualidade do teste para determinar se a meta é fazer o teste mais rápido, mais barato ou com uma cobertura maior.

Quais os processos de teste são melhores na necessidade de melhoria, quanto tempo os processos de melhoria podem durar e com qual esforço;

- **Determinar a situação corrente**: pontos fortes e fracos da situação corrente são determinados;
- **Determinar a situação requerida**: as ações necessárias para se chegar a situação requerida são determinadas;
- **Implementar mudanças**: a implementação é feita de acordo com um plano e situações são checadas para verificar que as metas foram alcançadas.

No modelo TPI é observado um processo de teste através dos diferentes pontos de vista, como: o uso de ferramentas de teste, técnicas de especificação de teste e relatórios. Estas são chamadas de áreas chaves no modelo TPI, onde cada área é classificada dentro de um nível de maturidade. Existem algumas dependências entre as áreas chave e os níveis. Por isso, uma matriz de maturidade de teste é usada. As classificações dos níveis são feitas através de pontos de checagem.

A metodologia TMap, para teste estrutural, é usada como base para o modelo TPI. A metodologia TMap contém quatro alicerces: o ciclo de vida (L) das atividades relacionadas ao ciclo de desenvolvimento, boa Organização (O), a correta infra-estrutura e ferramentas(I), e técnicas usáveis para realização das atividades (T).

Dentro destes alicerces um total de 20 áreas chaves podem ser reconhecidas para o modelo TPI. Estas áreas chaves cobrem o processo total de teste. Para permitir uma percepção no estado das áreas chaves, o modelo as fornece com níveis ascendentes, geralmente de A a D. Em média existem de 3 a 4 níveis por área chave. Cada nível mais alto é melhor do que o anterior em termos de tempo, custo e qualidade. A Quadro 3-1 representa os diferentes níveis das áreas chave, incluindo os alicerces.

As escalas da maturidade do teste pode ser divididas dentro de 3 categorias: Controlada (de 1 a 5), eficiente (de 6 a 10) e otimizada (de 11 a 13), o nível 0 é reservado para processos totalmente imaturos.

Devido ao grau de importância diferenciada entre cada área chave e nível associado, foi criada a Matriz de Maturidade de Teste que relaciona cada uma delas. A estrutura da Matriz de Maturidade de Teste é descrita na Quadro 3-1.

Quadro 3-1 - Níveis de maturidade TPI.

| | Áreas chave/Escala | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | Estratégia de Teste | | A | | | | | B | | | | C | | D | |
| 2. | Modelo de Ciclo de Vida | | A | | | B | | | | | | | | | |
| 3. | Momento de Envolvimento | | | A | | | | B | | | | C | | D | |
| 4. | Estimativa e Planejamento | | | | A | | | | | | | B | | | |
| 5. | Especificação de Técnicas de Teste | | A | | B | | | | | | | | | | |
| 6. | Técnicas de Teste Estáticas | | | | | A | | B | | | | | | | |
| 7. | Métricas | | | | | | A | | | B | | | C | | D |
| 8. | Automação de Testes | | | | A | | | | B | | | C | | | |
| 9. | Ambiente de Teste | | | | A | | | | B | | | | | | C |
| 10. | Ambiente de Trabalho | | | | A | | | | | | | | | | |
| 11. | Motivação e Comprometimento | | A | | | B | | | | | | | C | | |
| 12. | Funções de Testes e Treinamento | | | | A | | | | | | C | | | | |
| 13. | Escopo da Metodologia | | | | | A | | | | | | B | | | C |
| 14. | Comunicação | | | A | | B | | | | | | | C | | |
| | Áreas chave/Escala | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 15. | Reportar Erros | | A | | | B | | C | | | | | D | | |
| 16. | Gerenciamento de | | A | | | | B | | C | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Defeitos | | | | | | | | | | | | | | |
| 17. | Gerenciamento de Recursos de Teste. | | | A | | | B | | C | | | | D | | |
| 18. | Gerenciamento de processo de teste | | A | | B | | | | | | | | C | | |
| 19. | Avaliação | | | | | | | A | | | B | | | | |
| 20. | Baixo nível de teste | | | | | A | | B | | C | | | | | |

## 3.5 Modelos de Maturidade de Teste de Software

Nesta sessão será apresentada alguns modelos e frameworks de teste geralmente usados como base para definição de processo de teste de software, onde são instanciados e adequados de acordo com a necessidade da organização.

### 3.5.1 **Modelo 3P x 3E**

Esta metodologia proposta por Filho, 2002, foi proposta com o intuito de servir de modelo básico para elaboração, acompanhamento e monitoração do processo de teste de maneira aderente a Metodologia de desenvolvimento da organização que a utiliza.

Está metodologia faz uma abordagem onde o processo de teste deve ser inserido na primeira fase do desenvolvimento e serem realizados de maneira simultânea (paralela) até a implantação do software para que sejam atingidos os resultados esperados.

Segundo o modelo proposto, o processo de teste executa quatro etapas de maneira sequencial e outras duas etapas em paralelo como mostra a Figura 3-11.

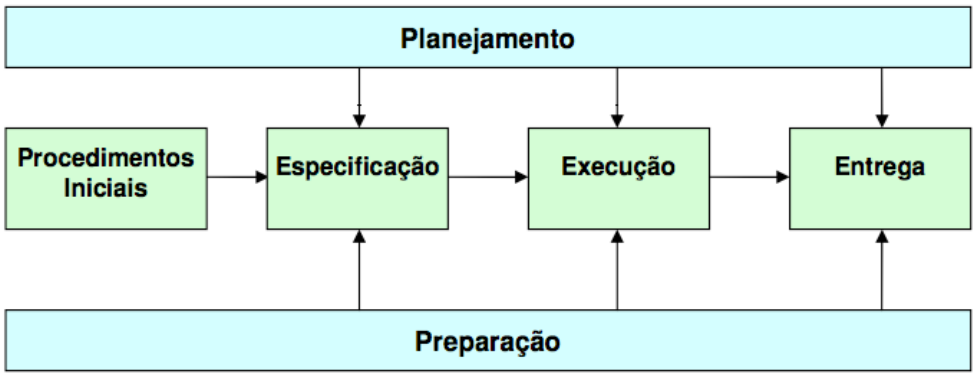


Figura 3-11 –Representação do Modelo “Y” de Teste (Molinari, 2006).

O modelo recebe o nome de 3E x 3P pela relação entre as atividades, produtos e documentos. O primeiro P (Procedimentos Iniciais) é uma fase relativamente pequena em que é traçado de maneira geral um pequeno esboço do processo de teste e assinado um acordo de nível de serviço. Os outros dois P's (Planejamento e Preparação) são atividades complementares que dão suporte ao processo e que acompanham todo o processo. Os E's (Especificação, Execução e Entrega) representam o núcleo do processo de testes e consomem em torno de 80% à 85% de todo o processo (Filho, 2002).

A execução das etapas seguem a seguinte seqüência:

- Procedimentos iniciais;
- Planejamento e Preparação;
- Especificação;
- Execução;
- Entrega.

As etapas do modelo são compostas por subetapas, insumos e produtos como mostra o Quadro 3-2

Quadro 3-2 - Etapas e Subetapas do Modelo 3P x 3E (Filho, 2002).

| Etapa | Subetapa | Insumo | Produto |
|---|---|---|---|
| 1. Procedimentos Iniciais | 1.1. Elaborar Guia Operacional de Teste | Requisitos do negócio;<br>Modelos de Dados;<br>Diagrama de fluxo de Dados;<br>Outros documentos de desenvolvimento. | Guia Operacional de Teste - GOT |
| 2. Planejamento | 2.1. Estabelecer Estratégias de Teste | Requisitos do negócio;<br>Modelos de Dados;<br>Diagrama de fluxo de Dados;<br>Outros documentos de desenvolvimento;<br>Guia Operacional de Testes. | Estratégia de Teste<br>Análise de riscos do Plano de Teste |
| | 2.2. Estabelecer Plano de Teste | Estratégia de Teste;<br>Análise de riscos do projeto de teste;<br>Necessidade de dados de teste;<br>Planejamento do sistema que está desenvolvendo. | Plano de Teste;<br>Análise de Riscos do projeto de testes |
| | 2.3. Revisar Estratégia de Testes | Requisitos de negócio do sistema;<br>Estratégia de teste; | Estratégia de testes revisada. |
| | 2.4. Revisar Plano de Testes | Estratégia de Teste; | Plano de Teste |

| | | | |
|---|---|---|---|
| | | Plano de Teste. | revisado. |
| **Etapa** | **Subetapa** | **Insumo** | **Produto** |
| 3. Preparação | 3.1. Adequar o Plano de Testes a Gerência de Configuração e/ou de Controle de Mudanças | Arquitetura do ambiente de desenvolvimento;<br>Arquitetura do ambiente de produção;<br>Ferramentas e procedimentos de Gerência de Configuração e de mudanças. | Registro e controle das diversas versões do produto: funcional, desenvolvimento, produto e operacional. |
| | 3.2.Disponibilizar Infra-estrutura e Ferramentas de Teste. | Estratégia de Testes;<br>Arquitetura Básica do ambiente de produção;<br>Ferramentas de Teste. | Infra-estrutura e ferramentas de teste disponíveis para a equipe de teste. |
| | 3.3.Disponibilizar Pessoal | Estratégia de Teste;<br>Plano de Teste;<br>Ferramentas de teste;<br>Definição do Ambiente de Teste. | Equipe de Testes definida e capacitada. |
| 4. Especificação | 4.1.Elaboração dos Casos de Teste | Estratégia de testes;<br>Plano de Teste;<br>Documentação técnica do sistema;<br>Necessidade dos dados para teste;<br>Posição quanto aos testes já realizados. | Casos te testes;<br>*Scripts* de testes;<br>Especificação das necessidades de dados de teste. |
| | 4.2.Elaborar Roteiros de Teste | Casos de Testes;<br>Planos de Teste;<br>Fluxo de execução dos programas previstos pela equipe de desenvolvimento. | Roteiros de Teste. |
| 5. Execução | 5.1.Preparar Dados de Teste | Casos de Teste;<br>*Scripts* de Teste;<br>Roteiros de Teste;<br>Documentação do sistema;<br>Especificação das necessidades de dados do teste;<br>Processos de criação de bases e/ou arquivos de teste. | Bases/Arquivos de teste disponíveis. |
| | 5.2.Executar Testes | Roteiros de teste;<br>Casos de teste;<br>*Scripts* de teste;<br>Resultados esperados. | Resultados dos testes;<br>Relatório de defeitos encontrados;<br>Ajustes no Matéria de testes. |
| | 5.3.Solucionar ocorrências de testes | Relatórios de defeitos com *status* a resolver;<br>Resultados de Testes. | Relatórios de defeitos encontrados com *status* resolvido ou a avaliar. |
| | 5.4.Acompanhar execução dos casos de testes | Relatórios de defeitos(Resumo);<br>Resultados dos testes(Resumo);<br>Estratégia de testes;<br>Plano de testes;<br>Casos de testes;<br>Relatórios de testes. | Análise do andamento dos casos de teste. |
| | 5.5.Elaborar relatório final | Análise dos relatórios de teste;<br>Estratégia de Testes;<br>Resultados de testes;<br>Relatórios de defeitos(resumo);<br>Plano de testes. | Relatório final dos testes. |
| 6. Entrega | 6.1.Avaliação e arquivamento da documentação. | Documentos de testes. | Relatório de não conformidade;<br>Relatório final de testes;<br>Documentação |

| | | | |
|---|---|---|---|
| | | | arquivada. |

Os principais documentos produzidos nesta metodologia são: GOT (guia operacional de testes), estratégias de testes, plano de testes, casos de testes e roteiros de testes.

Segundo, Filho, 2002, o GOT é um documento que não necessariamente precisa fazer parte do projeto, visto que é um documento com intuito de formalizar entre as partes envolvidas (desenvolvedores, usuários e equipe de teste), o início do projeto de testes, a responsabilidade de cada um e uma análise inicial dos riscos envolvidos.

Ainda segundo o Filho, 2002, o documento de estratégia de testes tem por finalidade fornecer uma visão geral do projeto de teste e as diretrizes para execução do processo. De acordo com as necessidades da utilização nas organizações estes documentos poderão ser alterados. Dentre as principais informações contidas na estratégia de teste, deve estar claramente definido o objetivo e as metas a serem atingidas, o ambiente de teste, tais como equipamentos, softwares, pessoal e ferramentas, e a abordagem dos testes a se considerar (unidades, integração, sistema e aceitação).

O plano de teste contém o projeto ou desenho lógico do processo de teste e está alinhado com a estratégia de teste. Define os objetivos gerais esperados e as expectativas do projeto de teste.

O documento caso de teste contém os testes propriamente ditos, detalhando campos de formulários, arquivos, telas e outros. As principais informações neste documento são: entradas e resultados esperados. No roteiro de teste deve ser registrada a sequência lógica de passos a ser executado nos casos de testes.

Em correspondência entre as etapas de desenvolvimento do sistema e a integração como os tipos de testes, a execução do teste não segue uma relação unívoca com as etapas de desenvolvimento, pois cada teste poderá corresponder a mais de uma etapa, figura 3.12 Esse modelo acaba sendo uma abstração do modelo em V (Sommerville ,2007), onde o processo de teste ocorre em paralelo com o processo de desenvolvimento desde o seu início.

O objetivo maior de se executar os testes cada vez mais próximos do início do desenvolvimento é a redução de custos (Filho ,2002). O custo da correção dos defeitos cresce exponencialmente à medida que o projeto de desenvolvimento avança dentro do ciclo de vida do sistema (Myers ,1979).

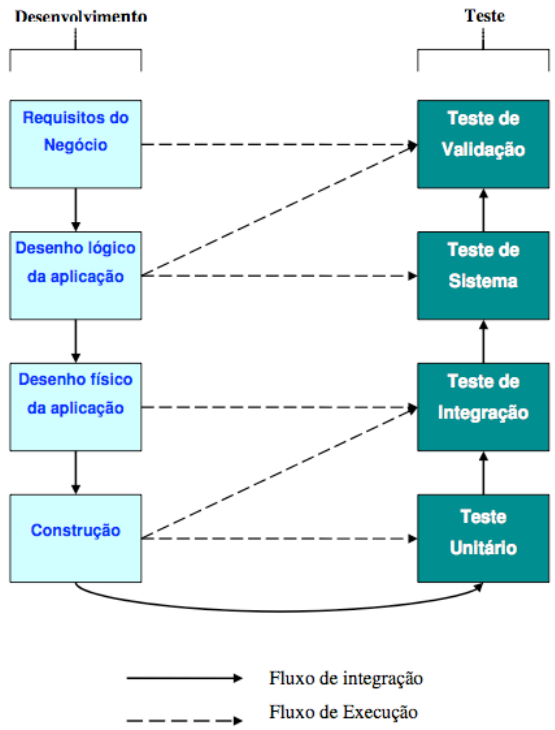


Figura 3-12 –Representação do Modelo "Y" de Teste (Molinari, 2006).

### 3.5.2 **Norma IEEE 829**

A Norma IEEE 829 tem como objetivo descrever uma série de documentos a serem utilizados nas atividades de teste de produtos de software. São descritos um total de oito documentos que cobrem as tarefas de Planejamento, especificação e relato de testes (IEEE 829, 1998):

- **Plano de Teste**: apresenta o planejamento para execução do teste, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste. Identifica também os itens e as funcionalidades a serem testados bem como as tarefas a serem realizadas e os riscos associados com as atividades de teste;
- **Especificação do Projeto de Teste**: refina a abordagem apresentada no Plano de Teste e identifica as funcionalidades e características a serem testadas pelo projeto e por seus testes associados. Este documento também identifica os casos e os procedimentos de teste se existirem, e apresenta os critérios de aprovação;
- **Especificação de Casos de teste**: define os casos de teste, incluindo dados de entrada, resultados esperados, ações e codificações gerais para a execução do teste;
- **Especificação de Procedimento de Teste**: especifica os passos para executar um conjunto de casos de teste;

- **Diário de Teste**: apresenta registros cronológicos dos detalhes relevantes relacionados com a execução dos testes:
- **Relatório de Incidente de Teste:** documenta qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior;
- **Relatório-resumo de Teste**: apresenta de forma resumida os resultados das atividades de teste associada com uma ou mais especificações de projeto de teste e provê avaliações baseadas nesses resultados:
- **Relatório de Encaminhamento de Item de Teste**: identifica os itens encaminhados para teste no caso de equipes distintas serem responsáveis pelas tarefas de desenvolvimento e de teste.

A norma separa as atividades de teste em três: preparação dos teste, execução do teste e registro do teste. A Figura 3-13 mostra os documentos que são produtos da execução de cada uma das fases e os relacionamentos entre eles.

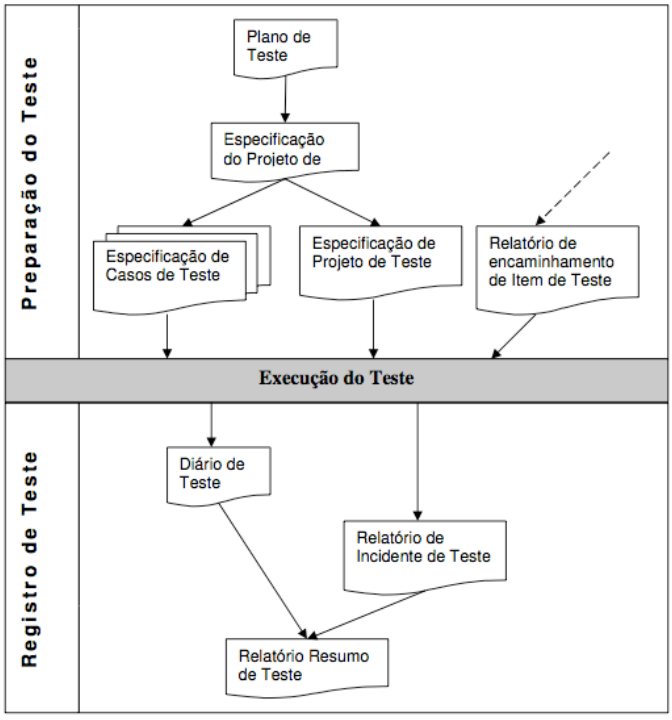


Figura 3-13 –Representação do Modelo "Y" de Teste (Molinari, 2006).

Adicionalmente, a equipe responsável pelo teste deverá tomar outras decisões em relação à aplicação da norma em projetos específicos, decidindo, por exemplo, se é mais conveniente elaborar um único plano que englobe os testes de unidade, integração e aceitação, ou um plano para cada uma das fases de teste citadas.

Mais do que apresentar um conjunto de documentos, a serem utilizados ou adaptados para determinadas empresas ou projetos, a norma apresenta um conjunto de informações necessárias para o teste de produtos de software. Sua correta utilização auxiliará a gerência a se concentrar tanto com as fases de planejamento e projeto quanto com a fase de realização de testes propriamente dita, evitando a perigosa armadilha de só iniciar a pensar no teste de um produto de software após a conclusão da fase de codificação.

### 3.5.3 **Modelo CenPRA**

A metodologia desenvolvida pelo então CenPRA(Centro de Tecnologia da Informação Renato Archer) , hoje CTI, é fundamentada na adoção de um processo de teste utilizando modelos sugeridos pela Norma IEEE 829, e tem como objetivo melhorar o processo de teste de softwares nas empresas através de técnicas, procedimentos e ferramentas. Desta forma, as empresas seriam capazes de desenvolver produtos de melhor qualidade (Csbsaj, 2004).

A metodologia é aplicada a qualquer tipo de software, seja ele sistema de informações ou software científico, e também podendo ser empregada tanto em empresas que desenvolvem quanto empresas que adquirem software.

A implantação do processo de teste envolve um conjunto de atividades que vai desde o levantamento das necessidades da empresa, passa pela realização de treinamentos da equipe técnica e vai até ao acompanhamento dos trabalhos realizados, constituindo assim, um completo ciclo de implantação da atividade de teste dentro da empresa.

O processo de teste proposto na metodologia está baseado em alguns pressupostos básicos:

- Os testes de sistemas e aceitação são projetados sob a responsabilidade da equipe de teste;
- Os testes de sistema e eventualmente de aceitação são realizados de forma iterativa, havendo, antes do início de cada ciclo de teste uma avaliação rápida do produto.

Os dois documentos que formam a base da metodologia são: Guia para Elaboração de Documentos de Teste de Software (CenPRA, 2001a) e Processos para Elaboração de Documentos de Teste de Software (CenPRA, 2001b). O primeiro tem o propósito de servir de referência para a criação dos documentos de teste tanto na fase de preparação para a atividade de teste quanto na fase de registro dos resultados do teste. O segundo documento, apresenta os processos relativos à preparação, execução e o registro dos testes.

Esses processos estabelecem uma orientação geral e, se necessário, podem ser modificados para adequar-se às situações particulares de organizações envolvidas nas atividades de teste. Um processo é definido para cada documento da Norma, de acordo com a seguinte estrutura (Csbsaj, 2004):

- **Funções e responsabilidades no processo**: participantes na execução das tarefas, divididos por funções e responsabilidades;
- **Critérios para início do processos**: elementos e/ou condições necessárias para iniciar a execução das tarefas;
- **Entradas do processo:** dados, recursos ou ferramentas necessárias para a execução das tarefas;
- **Tarefas do processo:** ações necessárias para produzir as saídas do processo. Para cada tarefa são identificados suas entradas, com identificação de possíveis fontes, e as saídas produzidas. A ordem de apresentação das tarefas não reflete necessariamente a sequência em que devem ser executadas;
- **Saídas do processo:** dados ou produtos gerados pela execução das tarefas;
- **Critérios para termino do processo:** elementos ou condições necessárias para encerrar a execução das tarefas;
- **Medições do processo:** medidas a serem coletadas como parte da execução das tarefas.

## 3.6 Considerações Finais

Como foi apresentado neste capítulo, para se obter qualidade em um produto de software é importante que a atividade de Teste de Software seja realmente "levada a sério", ou seja, que a organização que pretenda executar a atividade realmente disponibilize os recursos necessários como: equipe qualificada, infra estrutura, processo adequado e que seja planejada e não somente executada.

Não basta uma organização que deseje executar um teste de software efetivo, apenas copie um processo ou metodologia de teste, é importante que haja uma equipe qualificada para a atividade e que a metodologia ou processo seja aderente/adaptada ao processo de desenvolvimento. Caso os processos de desenvolvimento e de teste não sejam aderentes (que tenham uma convivência construtiva), a atividade de teste será tida apenas como destrutiva e não como construtiva, ou seja, será vista apenas como a etapa em que atrasa e não como uma etapa que imprima qualidade de ganhos para organização.

Através do Teste de Software associado as outras atividades de qualidade como Garantia e Controle a organização conseguirá atingir a qualidade exigida pelo cliente para seus produtos, melhorando sua visão interna, externa e perante ao cliente.

# Capítulo 4 - **Melhoria no Processo de Software no Contexto de Teste de Software**

Este capítulo, primeiramente apresenta o cenário onde o estudo de caso ocorreu, após a apresentação do cenário é apresentado o processo de desenvolvimento de software e o processo de Teste de Software Versão 1 utilizados antes das melhorias. Após a apresentação da primeira versão do processo de Teste de Software e com base na análise de pontos fortes, pontos fracos e oportunidades de melhorias, é apresentado o processo de Teste de Software Versão 2. Ao final do capítulo é realizada uma comparação entre o processo de Teste de Software da Versão 2 e outros processos de testes que foram levantados para avaliar a completitude e aderência aos padrões mercadológicos e científicos.

## 4.1 Contextualização do cenário do projeto

Esta seção contextualiza quanto ao cenário em que ocorre o estudo de caso e será divido em três subseções. A primeira descreve características sobre o fornecedor do produto, a segunda relata a descrição do Cliente e, por fim, a terceira ambienta sobre o projeto que foi contratado.

### 4.1.1 **Fornecedor**

A empresa foco deste estudo foi fundada em 1974 e foi a primeira empresa genuinamente nacional a produzir e comercializar tecnologia no segmento de informática. Nos anos 90, já fazendo parte da estrutura de um banco nacional, tornou-se uma integradora de solução. No ano de 2003, reforça sua posição como integradora e investe para se tornar a maior empresa do segmento no mercado nacional e com atuação no exterior (Cobra, 2011).

A missão do fornecedor é "Gerar valor para seus acionistas, colaboradores e a sociedade, provendo seus clientes com soluções em tecnologias da informação que sejam ambientalmente sustentáveis e socialmente sustentáveis". Possuindo sua matriz na cidade do Rio de Janeiro, a organização possui mais 28 CAT (Centro de Atendimentos Tecnológicos) espalhados no Brasil.

Já no ano de 2004 iniciou o desenvolvimento de um Programa de Excelência Tecnológica (PET) em função da necessidade de atendimento a um cliente, com foco na execução dos seguintes serviços: Operação Assistida, Suporte, Redes, Desenvolvimento de Software e Gerência de Projetos (Cobra, 2011).

Em 2007, foi criado um projeto piloto de Fábrica de Software [5], através da contratação de um gerente de projetos com a responsabilidade de estruturar e gerenciar uma Fábrica de software, que, então passou a fazer parte do seu portfólio de serviços.

Atualmente, a Fábrica conta com aproximadamente 140 (cento e quarenta) colaboradores com papéis definidos (com responsabilidades e atribuições especificados de acordo com a matriz de competências definida pela organização) e trabalhando em processo também definido e institucionalizado, divididos em diferentes áreas de atuação como: Negócio, Análise de Requisitos, Desenvolvimento de Software e Qualidade de Software.

Com o incremento da atividade de desenvolvimento de software em seu portfólio, o fornecedor passou a oferecer 15 serviços e/ou soluções, são eles (Cobra, 2011):

- Assistência Técnica;
- *Contact Center*;
- Produção de Documentos;
- Gerenciamento Eletrônico de Documentos;
- Institucional;
- Fábrica de Software;
- Soluções em Software Livre;
- Migração para Software Livre;
- Solução para *Data Center*;
- Produção de Documentos;
- Central de Comunicação em VOIP;
- Gerenciamento Escolar;
- Solução integrada de Gestão Municipal;
- *Máquinas Desktop*;
- *Máquinas Notebooks*;

[5] Fabrica de Software é um conjunto de Recursos (Humanos e Materiais), Processos e Metodologias estruturados de forma semelhantes aquelas das industrias tradicionais utilizando as melhores praticas para o processo de desenvolvimento, testes e manutanção dos softwares(Pressman, 2007)

- *Máquinas Servidores*;

### 4.1.2 **O cliente**

Fundado em 1942, a partir de um acordo entre Brasil e Estados Unidos, criou-se o Decreto-lei de número 4.451, em 9 de julho, fundando o então Banco da Borracha. Já no ano de 1950 com a ampliação das linhas de crédito para outras áreas, o governo federal cria o Banco Beta (Nome fictício dado para o cliente). No século 21, o Banco Beta possui sua atividade voltada para o Desenvolvimento Sustentável da Amazônia Legal, através da definição de critérios rigorosos na análise do crédito.

O Banco Beta é guiado pela seguinte missão: "prover o desenvolvimento da região amazônica desempenhando um papel importante tanto no âmbito da pesquisa, quanto no crédito de fomento". O cliente é responsável por 60% dos créditos de longo prazo da região e está presente na Amazônia legal e mais os Estados de São Paulo, Distrito Federal e Rio Grande do Sul.

Para que a instituição consiga alcançar e desempenhar sua missão, desde 2004 passa por uma atualização do parque tecnológico para que consiga atender aos seus paradigmas de negócio, de maneira a aprimorar os níveis de competitividade e qualidade de serviços e, com isto, viabilizarem as diretrizes do Planejamento Estratégico da Organização.

### 4.1.3 **Projeto**

No ano de 2007, é iniciado o primeiro e maior projeto em desenvolvimento do PET, o Sistema de Gestão de Fomento. Este consiste, basicamente, na administração da captação de recursos de fundos externos pelo Banco e a gerência da destinação destes recursos, por meio dos contratos de fomento rural, industrial, comércio e serviços, firmados entre o Banco e os proponentes. O sistema de Gestão de Fomento obteve seu tamanho estimado, inicialmente, em 4560 UCPs (Pontos de Caso de Uso). Porém, com a identificação de novas necessidades ao longo do projeto, o número de UCPs encontra-se atualmente em revisão.

O sistema é dividido em quatro grandes entregas, onde cada uma possui um conjunto de funcionalidades ou parte deles em condições de entrar em produção e agregar valor ao cliente. A primeira consiste no sistema ser capaz de realizar o acolhimento da proposta do cliente, realizando a reserva de dotação de recursos. A segunda entrega trata da alçada, que visa realizar a decisão e análise quanto à proposta acolhida, com base na dotação. A terceira entrega consiste nas propostas acolhidas gerarem contratação para os clientes. E, por fim, na quarta entrega são realizados os Cálculos e ré-cálculos de destinações.

## 4.2 Processo de Teste de Software Versão 1

Esta seção descreverá primeiramente o cenário em que o processo de Teste de Software Versão 1 era executado, após será apresentado o processo de Desenvolvimento de Software do Produto, os papéis participantes e os artefatos que faziam parte do processo e o detalhamento e atividades constantes no processo.

### 4.2.1 **Cenário de aplicação do Processo de Teste de Software**

Em função da alta complexidade dos requisitos, pouco tempo destinado ao desenvolvimento do projeto e escopo grande, apresentou-se uma constante necessidade de comunicação entre a equipe executora do projeto com diversos *stakeholders* do cliente. Com base nessa necessidade, foi acordado que a fábrica ficaria sediada no mesmo espaço físico do cliente, de maneira a facilitar, principalmente, a atividade de Elicitação de Requisitos. Acordou-se, ainda, que o sistema seria entregue por etapas (entregas funcionais), como discutido na seção 4.1.3. Foi acordado, também, que a entrega ficaria por um período em homologação, para que o cliente pudesse realizar as validações dos requisitos implementados e só, então, ser implantado em ambiente de produção.

Tratando do cenário do projeto de Gestão de Fomento, o fornecedor enfrentou diversas dificuldades quanto à aquisição de recursos. O projeto não contemplava disponibilização de recursos de software e hardware. Com isso, o projeto foi direcionado a usar os recursos de software em que o cliente dispusesse a licença de uso, para economizar recursos e direcioná-los para compra de melhores hardwares. Com esta limitação, diversas ferramentas de software livre foram usadas para apoiar diferentes disciplinas da Engenharia de Software, como exemplo tem-se a adoção do sistema operacional Linux, ferramenta de automação de testes Selenium, JMeter, entre outros.

No início do projeto, a equipe era composta pelos seguintes perfis: 2 Gerentes de Projeto (dividindo as atribuições no projeto), Analistas de Requisitos, Arquitetos de Software, Projetista de software, Administrador de Dados e Desenvolvedores, sendo um dos desenvolvedores com responsabilidade de definição dos padrões gráficos do sistema.

A participação do cliente resumia-se em ser fonte de conhecimento do negócio e de consulta sobre os requisitos funcionais (área gestora do negócio), validação de requisitos e na homologação (aceitação) do sistema, ao invés do acompanhamento geral do projeto. Isso gerava, na fase de Homologação, novas discussões sobre o negócio e ocasionava mudanças.

Como a área do cliente responsável pelos Projetos de T.I, geralmente não havia conhecimento de novas discussões e sobre o negócio e acertos quanto as mudanças, os prazos eram apenas estimados e cobrados contratualmente.

A ferramenta de gestão de defeitos era utilizada apenas para registro de defeitos em fase de homologação (registro do cliente), mudanças e registro de tarefas (registro realizados pelo gerente de projetos referente a tarefas da fábrica). Deixando, assim, os registros de defeitos internos sem gestão de ferramentas, apenas no conhecimento dos colaboradores que relatavam e corrigiam esses defeitos.

Os recursos de hardware fornecidos pelo cliente não eram adequados as atividades desempenhadas pela equipe, uma vez que esta atividade de desenvolvimento de sistemas exige altas capacidades de máquinas.

### 4.2.2 **Visão Geral do Processo de Desenvolvimento de Software**

Esta seção descreve primeiramente o Processo de Desenvolvimento de Software, posteriormente é dado ênfase no processo de Teste de Software de Versão 1 do processo em questão. Para descrição dos processos serão utilizadas notações do BPMN (*Business Process Modeling Notation*) (BPMN, 2011) a fim de facilitar o entendimento e melhor expressividade das notações de processo.

O Processo de Desenvolvimento de Software é basicamente dividido em sete macro-atividades: Requisitar, Projetar Software, Gerar Banco de Dados, Desenvolver Software, Testar Software, Homologar Software e Gestão; como mostra a Figura 4-1.

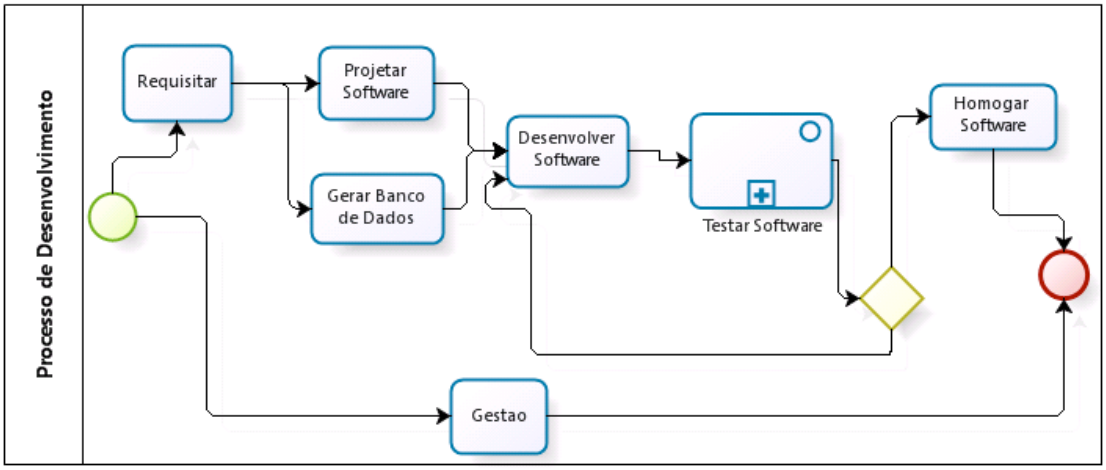


Figura 4-1 –Processo de desenvolvimento de software.

A primeira macro-atividade do Processo de Software antes das melhorias é Requisitar, que consiste nos Analistas de Requisitos elicitarem as necessidades do cliente através de reuniões e entrevistas, que são registradas em Atas, e as representam em Regras de Negócios. Após a construção do documento de Regras de Negócio, o Analista de Requisito produz a documentação de Casos de Uso, que consiste em uma documentação que aproxima a Regra de Negócio da linguagem de programação.

Ao término da atividade de Requisitar, as macro-atividades Projetar Software e Modelar Bando de Dados ocorrem em paralelo. A atividade de Projetar Software, consiste no Projetista de Software projetar o software através de especificação de diagramas e modelos de projeto usando o padrão da UML (*Unified Modeling Language*), gerando a Especificação de Análise (EA), recebendo como insumo o Caso de Uso e Regras de Negócio desenvolvidos/descritos pelos Analistas de Requisitos. A atividade de Modelar Banco de Dados consiste no Administrador de Dados modelar a Base de Dados, tomando como base os Casos de Uso, e gerar a documentação de Dicionário de Dados de maneira a atender as necessidades do negócio e fornecer subsídios ao Desenvolvedor que será responsável pela macro-atividade de Desenvolver Software.

A macro-atividade Desenvolver Software recebe como insumo o Caso de uso, Regras de Negócio, Especificação de Análise, Dicionário de Dados e Base de Dados modelada, para, então, o Desenvolvedor realizar as atividades de Desenvolver Software.

No Processo de Desenvolvimento de Software, só após a atividade de Codificar Software é que se inicia as atividades de Teste de Software. Neste processo ocorrem duas atividades de teste em paralelo: a atividade de Elaborar Projeto de Teste de Software e Executar Testes de Software. Ao encontrar defeitos, os mesmos eram informados aos Desenvolvedores que, então, os analisavam e corrigiam.

As atividades Elaborar Projeto de Teste de Software e Executar Teste de Software, neste processo, eram desempenhadas pelo mesmo Analista de Requisitos responsável pela atividade de Requisitar. As atividades eram executadas em paralelo, ou seja, ao mesmo tempo em que o Analista de Requisitos escrevia os casos de teste[6], executava-os.

---

[6] Casos de Teste: Conjunto de condições utilizadas para Teste de Softwares Ele pode ser elaborado para identificar defeitos na estrutura interna do software, através de situações que exercitem adequadamente todas as estruturas utilizadas na codificação (Molinari, 2003).

A última macro-atividade do Processo é a atividade Homologar Software, que descreve o fato do software ser disponibilizado em um ambiente próprio do cliente para que o produto pudesse ser verificado e validado, a fim de receber a homologação (aceitação). Quando eram encontrados defeitos e problemas em fase de homologação, os mesmos eram registrados na ferramenta de *bugtracking*[7] e informados ao fornecedor do produto que, então, era dado o encaminhamento para serem corrigidos. Durante este processo, a fase de homologação era conduzida pelos Analistas de Requisitos que haviam participado da análise do módulo e dos testes.

Para tratamento de defeitos na atividade de Homologar Software, foi acordado entre o fornecedor e o cliente um SLA (*Service Level Agreement* – Acordo de Nível de Serviço) para o projeto, onde foi definido que a criticidade dos defeitos (erro de funcionamento comum de um software) seria tratada como: crítico, importante e trivial. Para a homologação finalizar, o sistema: não poderia possuir nenhum defeito crítico (defeito que impede que o cliente realize sua tarefa) pendente; poderia apresentar menos de 25% de defeitos classificados como importantes (defeito que dificulta ou atrapalha a realização da tarefa, porém não a evita), onde o fornecedor mesmo assim possui o compromisso de corrigi-los, mesmo após o módulo ser homologado; e poderia ter defeitos triviais, os quais não seriam impeditivos para a homologação, devendo o fornecedor corrigir depois que módulo principal fosse homologado.

As mudanças eram registradas em uma ferramenta de *bugtracking*, onde, posteriormente, eram avaliadas por um Comitê de Gestão de Mudanças que era composto por representantes do cliente e do fornecedor para a tomada de decisão de quais mudanças seriam executadas.

### 4.2.3 **Processo de Testes de Software Versão 1**

Nesta seção será apresentado com mais detalhes o processo de Teste de Software da Versão 1 participante do processo de desenvolvimento. A Figura 4-2 mostra as atividades do Processo de Teste de Software Versão 1.

A atividade Projetar Teste de Software consiste no responsável, com base nos requisitos referentes as funcionalidades, projetar os Casos de Teste contendo os passos a

---

[7] Ferramenta de *Bugtracking* são ferramentas responsáveis pela gerência de registro de defeitos efetuados.

serem seguidos para seu atendimento, bem como dados de entrada e dados esperados de saída, dando origem ao artefato de Projeto de Teste.

Em paralelo, a atividade Executar Teste de Software é iniciada, onde nesta atividade ao mesmo tempo que os casos de testes vão sendo escritos, os mesmos são executados. Os resultados dos Casos de Testes vão sendo relatados no mesmo artefato em que foram projetados, relatando se os casos de teste foram satisfeitos ou não.

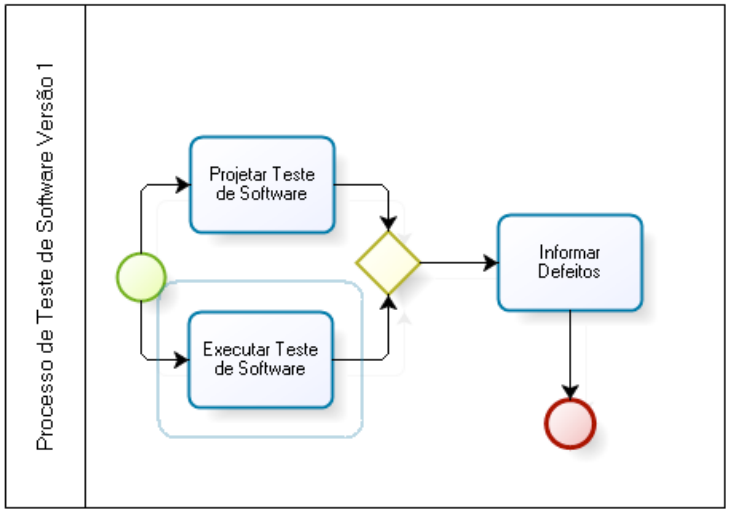


Figura 4-2 –Processo de Teste de Software Versão 1.

A atividade Informar Defeitos consiste na atividade de informar os defeitos encontrados durante os testes aos responsáveis pelas correções. Após o conhecimentos dos defeitos os responsáveis os corrigiam.

Os tipos de testes previstos e utilizados neste processo obtinham pouca completitude, pouca cobertura ao negócio e aos casos de uso. Os testes tinham como principal objetivo provar que o software funciona, enquanto que segundo Molinari (2003), o objetivo dos processos de testes mais maduros é de provar que o sistema faz o que tem que fazer e não faz o que não tem que fazer.

#### 4.2.3.1 Papéis utilizados no Processo de Teste de Software da Versão 1

Mostra quais são os papéis que diretamente se relacionam ao processo. Entende-se que papéis também são considerados funções que uma pessoa desempenha na empresa, sendo que uma pessoa pode realizar mais de um papel.

Os papéis participantes do processo foram divididos em duas partes: a primeira são descritos os papéis que apóiam a execução do processo de teste; enquanto que na segunda são

descritos os papéis que realmente fazem parte da execução do Processo de Teste de Software da Versão 1.

Os papéis que apóiam as atividades relacionadas ao processo de teste de software, mas não as realizam, são apresentadas no Quadro 4-1.

Quadro 4-1 - Papéis de apoio ao Processo de Teste de Software verão 1

| Papel | Descrição |
|---|---|
| Analista de Requisitos | Responsável por gerar artefatos de Casos de Uso, Regras de Negócio e modelagem UML. Tem como objetivo transferir o conhecimento do negócio para o restante da Fábrica de Software. |
| Analista Documentador | Responsável por elaborar a documentação técnica e funcional dos produtos da empresa (manuais de utilização/administração, etc.). |
| Gerente de Projeto | Responsável por realizar atividades relacionadas de gestão de escopo, negociação de prazos, e gestão de recursos humanos. |
| Analista de Dados | Responsável por atuar no projeto de análise do Modelo de Dados Lógicos e Físicos do sistema. |
| Arquiteto de Software | Responsável por especificar os padrões e metodologias que deverão ser utilizados pelo programador para implementar as regras de negócio, de forma a atender aos requisitos da empresa. |
| Desenvolvedor | Papel responsável por codificar e testar, de forma unitária, programas de computador, estabelecendo os processos operacionais necessários para o tratamento dos dados, baseando-se nas definições fornecidas na fase de análise de sistemas e valendo-se de métodos e técnicas adequadas aos equipamentos e aplicações a que se destinam. |

Já o Quadro 4-2 apresenta e descreve os papéis que participam efetivamente do processo de Teste de software.

Quadro 4-2 – Papéis que participem do processo de Teste de Software

| Papel | Descrição |
|---|---|
| Analista de Teste | Responsável por verificar se o produto está sendo desenvolvido de acordo com os requisitos, regras de negócio, padrão de interface e se o produto está funcionando de acordo com o esperado |

No processo de desenvolvimento de software antes das melhorias, não necessariamente um ator desenvolvia apenas um papel, as pessoas que desempenhavam a função de Analistas de Requisitos, por exemplo, desenvolviam papéis de Analistas de Testes e os Arquitetos de Software poderiam também desempenhar papéis de Desenvolvedor.

#### 4.2.3.2 Artefatos do Processo de Teste de Software Versão 1

Nesta seção são descritos os artefatos produzidos no processo de Teste de Software Versão 1. O Quadro 4-3 apresenta e descreve os artefatos de entrada ao processo de teste de

software, enquanto que o Quadro 4-4 apresenta e descreve os artefatos produzidos durante o processo.

Quadro 4-3 – Artefatos de entrada ao Processo de Teste versão 1.

| Artefato | Descrição |
|---|---|
| Regra de Negócio | Tem o objetivo de definir o comportamento do sistema, como as regras de negócio de uma determinada ação. Às vezes pode ser denominada especificação funcional. |
| Caso de Uso | Tem como objetivo prover a interação entre um ator e o sistema, ou ainda entre sistemas, ou entre hardware e sistema, mostrando as mensagens e o comportamento desta funcionalidade. |
| Modelo de Análise de Projeto | Tem como objetivo verificar possíveis inconsistências de requisitos e casos de uso. Neste artefato também é possível verificar as dependências entre partes do sistema. |
| Dicionário de Dados | Possui como objetivo definir todos os termos em comum utilizados no projeto, é equivalente a um dicionário que contem informações de conceitos, notações utilizadas e padrões. |
| Padrão de interface | Possui como objetivo descrever o funcionamento padrão dos componentes gráficos utilizados no sistema. |
| Documentação de Integração | Documentação que guarda detalhes sobre integrações externas entre o sistema desenvolvido e outros sistemas. |

Quadro 4-4– Artefatos produzidos pelo processo de teste versão 1

| Artefato | Descrição |
|---|---|
| Projeto de Teste | Documento responsável por armazenar casos de testes, os passos rapa reprodução e os resultados esperados. No processo em questão é de responsabilidade do Analista de Teste a geração e execução deste documento. |

## 4.3 Avaliação para Melhorias do Processo de Teste de Software da Versão 1

Após a execução do processo de Teste de Software da Versão 1, foi realizada uma reunião de retrospectiva com o objetivo de levantar os Pontos Fortes, Pontos Fracos e Oportunidades de Melhorias. Portanto, esta seção possui como objetivo apresentar os Pontos Fortes, Pontos Fracos e Oportunidades de Melhorias identificados na reunião de retrospectiva apresentada.

Para a identificação dos pontos fortes, pontos fracos e oportunidades de melhorias foi utilizado o Guia de Implementação do MPS.BR para as áreas de processo de Verificação e Validação. Com base no guia de implementação, através de uma análise, foram identificados pontos que o Processo de Teste de Software da Versão 1, necessitaria contemplar para se tornar aderente ao padrão MPS.BR para as áreas de VER e VAL.

A reunião de retrospectiva tinha como participantes os Analistas de Teste de Software, representantes das outras áreas (Desenvolvimento, Análise de Requisitos e Gestão) e o Coordenador da equipe de Teste.

Como pontos fortes foram destacados os pontos listados abaixo:

- Existência da atividade de Teste de Software no Processo – O fato da atividade de Teste de software existir no processo mostra que a organização possui o interesse em qualidade do produto, mesmo que com um processo imaturo, os testes são executados;
- Existência de um Documento de Projeto de Testes – A existência de um documento de projeto de teste de software é um ponto forte, uma vez que evidencia formalmente a atividade e o guarda para futuras execuções e/ou testes regressivos em casos de usos projetados, evitando assim retrabalho.
- Registro dos Defeitos Encontrados no documento de Projeto de Teste – O registro de resultados é um ponto forte, uma vez ficando evidenciado onde o software falhou, de maneira a poder contribuir com estatísticas e melhorias do processo de desenvolvimento, bem como guardar histórico de falhas.

Como pontos fracos destacam-se:

- As atividades de Teste de Software não são desempenhadas por profissionais dedicados e especializados em Teste de Software, o que acaba por deixar a atividade menos específica, podendo, assim, comprometer a qualidade da atividade e consequentemente a qualidade do produto;
- As atividades de Projetar Teste de Software e Executar Teste de Software ocorrem em paralelo, o que torna a atividade de projetar teste de software sujeita a falhas, uma vez não possuindo o tempo adequado para projetar os testes, podendo, assim, comprometer a completude da cobertura dos casos de uso trazendo como conseqüência perda de qualidade do produto e da própria atividade;
- Mesmo os defeitos sendo registrados no projeto de teste de software, juntamente com os casos de usos, a falta de uma formalização no repasse dos defeitos para os responsáveis pela correção, faz com que muitas vezes os defeitos fiquem sem registro de descoberta, procedimento de correções e o responsável pelas correções, gerando assim um desalinhamento entre a quantidade de defeitos encontrados e quantidade de defeitos corrigidos;
- Não há um subprocesso de Gestão de Defeitos, o que tornaria mais claro o processo de correção de defeitos, sua criticidade, seu tempo de correção, o responsável pela descoberta e pela correção e ainda quando o defeito estará disponível para novo teste;

- Como a atividade é executada por Analistas de Requisitos e os mesmos são comprometidos com a análise do requisito, o período de projetar e executar testes ficam muito curto. Em boas práticas de teste de software não é recomendado que nenhum outro profissional envolvido diretamente com o desenvolvimento do produto, seja o único responsável pelos testes, podendo ocasionar visões viciadas sobre o negócio e o produto e em função do tempo tornando a cobertura dos testes ao produto limitada;
- Apenas um tipo de teste é realizado, o funcional. Com a execução de apenas um tipo de teste torna a completude e cobertura dos testes pequena, de maneira a não confrontar o sistema com todos os artefatos como fariam outros tipos de testes;
- Uma das premissas básicas da atividade de teste é deixada de lado, Testar se o produto faz o que é para fazer e não faz o que não é para fazer – Em função do tempo curto, completude baixa e cobertura pequena, a abordagem utilizada para os testes é de testar apenas se o sistema esta fazendo o que é pra fazer ao invés de testar também se ele não faz o que não é pra ser feito, assim deixando de realizar testes em uma grande parte do sistema.

Algumas oportunidades de melhorias identificadas no processo:

- Criação dos papéis dedicados ao Teste de Software (Líder de Testes e Analistas de Testes);
- Definir onde a atividade de Projetar Teste de software será iniciada, bem como definir o momento que a atividade de Executar Teste de Software iniciará, uma vez essas atividades possuírem necessidades de início e termino em momentos diferentes dentro do processo e ainda uma gerando artefatos de entrada para a outra;
- Utilizar ferramenta de *bugtraking* para formalização do repasse dos defeitos e implementar um Processo de Gestão de Defeitos;
- Aumentar a cobertura dos testes quanto ao sistema, elaborando mais casos de testes;
- Adicionar os tipos de testes de Conformidade funcional, Integração e Controle de Acesso, para poder testar de maneira mais efetiva se o sistema está de acordo com as exigências negociais;

- Testar com o intuito de verificar se o software está fazendo o que deve fazer e se não está fazendo o que não deve fazer.

## 4.4 Processo de Teste de Software Versão 2

Esta seção contextualizará o processo de Teste de Software Versão 2, o processo que a organização passou a utilizar após a aplicação da avaliação realizada. Esta seção tratará do cenário em que o processo de Teste de software Versão 2 foi executado, descrevendo as atividades, os artefatos e os papéis, e ao final realizará uma análise do processo.

### 4.4.1 **Cenário de Aplicação do processo de Teste de Software Versão 2**

Na seção 4.3 foram apresentados pontos fracos, pontos fortes e oportunidades de melhorias que evidenciam que o processo de Teste de Software da Versão 1 estava requerendo aperfeiçoamentos. O objetivo de atender a essas melhorias é garantir uma maior qualidade ao produto desenvolvido mantendo os aspectos positivos, eliminando os pontos fracos e implementando oportunidades de melhorias.

O levantamento realizado abrangeu a estrutura organizacional, as ferramentas utilizadas, capacitação de pessoas através de treinamentos, *workshops* e o processo de teste de software, o qual será utilizado para uma análise de sua abrangência ao Modelo de Melhoria de Processo do Software Brasileiro (MPS.BR) e também a abrangência ao TMM (*Test Maturity Model*) afim de avaliar a aderência a esses modelos afim de verificar e aperfeiçoar a maturidade do processo.

Uma equipe específica para teste de software foi contratada e dividida em dois perfis: Líder de Teste e o Analista de Teste. O Líder de Teste é o perfil que dentro da equipe de teste de software é responsável por definições do processo de teste, *templates* de documentos, procedimentos, atividades, metodologias, atribuição/divisão das tarefas, remoção de impedimentos encontrados pela equipe além de reportar à gerência a situação da qualidade do produto e andamento das atividades. O perfil de Analistas de Teste possui como função elaborar projeto de teste de software e executar os testes de software, designado pelo Líder de Testes.

A equipe de desenvolvimento de software no cenário após as melhorias contava com 85 pessoas. Na equipe de teste de software 7 pessoas desempenhavam o perfil de Analistas de Testes e 1 de Líder de Teste.

Neste cenário a equipe do projeto e os clientes envolvidos foram alocados em uma única sala dedicada ao projeto com o intuito de facilitar a comunicação e preservar a privacidade do projeto. Os computadores inapropriados para determinadas funções como testes e desenvolvimento foram substituídos por computadores mais modernos e eficazes que atendiam a necessidade de cada área.

O cliente, havia sua participação em dois momentos, o primeiro na elicitação de requisitos contribuindo com o conhecimento do negócio (gestores do negócio) e o segundo momento na homologação do sistema, onde a equipe de T.I do cliente juntamente com a equipe de negócio realizavam testes de aceitação.

A relação entre o cliente e o fornecedor apresentavam algumas divergências quanto a homologação do produto, onde o cliente reclamava de o produto não atender suas expectativas e o fornecedor argumentava que os requisitos não eram estáveis e que o produto estava desenvolvido de acordo com os requisitos aprovados pelo próprio cliente.

Para o desenvolvimento do sistema eram utilizados três ambientes distintos, o ambiente de desenvolvimento, o ambiente de transição e o ambiente de homologação. Cada um com servidores de aplicação e banco de dados próprios. O ambiente de desenvolvimento consiste no ambiente que contém o sistema em constante desenvolvimento, já o ambiente de transição é um ambiente restrito apenas a equipe de testes, onde novas build´s (versões) são geradas apenas quando o coordenador de testes solicita como mostra a Figura 4-3.


Figura 4-3 – Ambiente utilizados para Desenvolvimento, Teste e Homologação.

Como mostra a Figura 4-4, os testes decorrem em ciclos de teste, onde cada ciclo contém rodadas de tipos de testes e estas rodadas representam a quantidade de vezes que um determinado tipo de testes foi repassado para uma determinada build. Um ciclo corresponde a uma Build. Como exemplo, poderia citar que em uma build X foi executado um ciclo de teste

Y, onde RF1 corresponde a uma rodada de testes funcionais da build X e RF2 a segunda rodada de testes funcionais, RA1 corresponde a uma rodada de teste de controle de acesso, ou seja, neste ciclo houveram duas rodadas de testes funcionais e mais uma de controle de acesso.

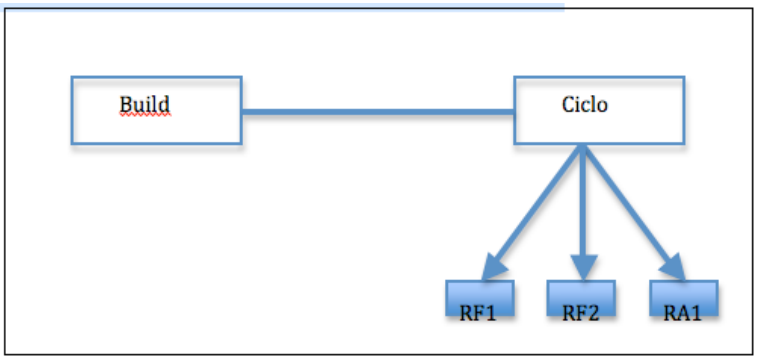


Figura 4-4 – Relação Ciclo de Teste x *Build* x Rodada de Teste.

### 4.4.2 **O Processo de Teste de Software Versão 2.**

Baseado nos pontos fracos, pontos fortes e oportunidades de melhorias apresentados na seção 4.3, foram realizadas as mudanças no processo de teste, cujo fluxo de atividades é descrito conforme a Figura 4-5 e detalhadas no Apêndice E.

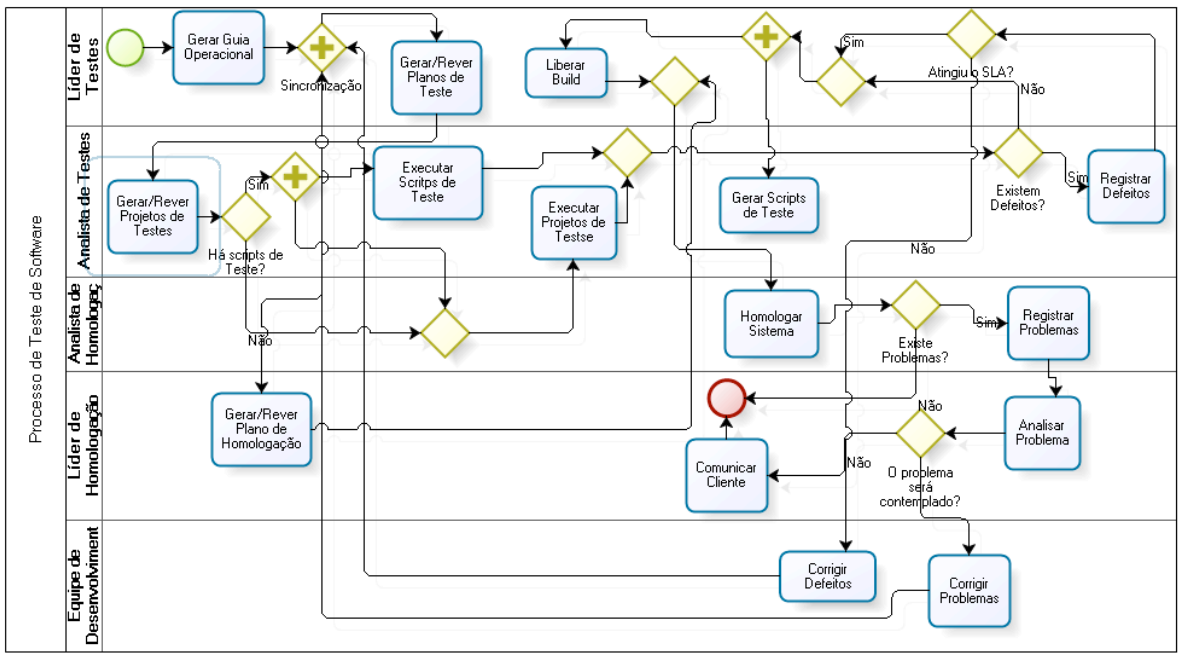


Figura 4-5 – Fluxo de Atividades no processo de Teste de Software de Versão 2.

A atividade Gerar Guia Operacional trata-se da fase desempenhada pelo Líder de Testes que possui como objetivo a geração de um documento que descreve as atividades que são realizadas ao longo da execução das atividades do processo de Teste de Software de Versão 2, e as responsabilidades de cada participante do processo. Após a atividade Gerar Guia Operacional outras duas atividades são disparadas em paralelo: Gerar Plano de Testes e Gerar Plano de Homologação.

A atividade Gerar/Rever Plano de Testes de responsabilidade do Líder de Testes possui como objetivo a geração de um documento com a função de elicitar e expressar as necessidades do teste do produto ou componente do produto, estabelecer a estratégia de testes que será usada para o sistema ou parte do sistema, estabelecer responsáveis e a geração do cronograma.

O Líder de Homologação é responsável pela atividade Gerar/Rever Plano de Homologação que possui como objetivo a geração de um documento que guia o cliente no processo de homologação do produto ou parte do produto. O plano de homologação além de guiar o cliente elenca as necessidades e insumos necessários para o bom funcionamento do sistema em ambiente de homologação.

A atividade Gerar/Rever Projeto de Teste que procede a atividade Rever/Gerar Plano de Teste, a atividade Projeto de Teste possui o objetivo de gerar o documento de projeto de teste que contém os casos de testes que descrevem os procedimentos a serem realizados e resultados esperados do teste para o produto ou parte do produto.

Após a atividade Geração e Revisão do Projeto de Teste, caso existam scripts, são executadas em paralelo duas atividades, a primeira é a atividade Executar Projeto de Teste, onde possui como objetivo a execução dos testes projetados na fase anterior. Além da execução dos projetos de testes essa fase é responsável pelo fechamento de ocorrências abertas, caso hajam.

A segunda atividade é a atividade Executar Scripts de Testes, onde, possui como objetivo a execução de scripts de testes regressivos afim de garantir que as mudanças não inseriram defeitos no sistema.

Após a execução dos testes, caso haja defeitos a atividade Registrar Defeitos, de responsabilidade dos Analistas de Testes é executada e possui como objetivo o registro de defeitos encontrados na atividade de execução dos testes. Caso não haja defeitos, a Build é liberada para a Homologação.

Após os Registros dos Defeitos uma checagem se mesmo com os defeitos registrados a *build* satisfaz ao SLA estabelecido entre cliente e fornecedor, caso atenda ao SLA em paralelo, são executadas a atividade Gerar *Scripts* de Testes e a atividade Entregar Build.

Na atividade Gerar *Scripts* de Teste os Analistas de Testes desenvolvem os Scripts de Testes, e os mesmos são guardados em um banco de scripts para utilizações em outros ciclos. A atividade de Liberar Build consiste em que a build ser liberada para a homologação, esta fase é de responsabilidade do Líder de Teste, através de um documento contendo, se houver, os defeitos conhecidos, e os testes realizados.

Caso o SLA não seja atendido, o fluxo segue para atividade Corrigir Defeitos, que possui o objetivo de a equipe de Desenvolvimento corrigir os defeitos relatados pela equipe de testes e disponibilizar uma nova build para que os testes sejam executados novamente.

A atividade Homologar Sistema possui como objetivo o acompanhamento dos "Testes de Aceitação" realizados pelo cliente e condução da homologação, com o objetivo final de receber o aceite do cliente quanto ao produto entregue. Caso haja problemas encontrados na atividade Homologar Sistema os mesmos devem ser analisados pela atividade Analisar Problemas, onde nesta atividade verifica-se se o problema realmente persiste, caso persista os mesmo são encaminhados atividade de Corrigir Problemas, caso não haja problemas o sistema recebe o aceite.

Um maior detalhamento das atividades com papéis responsáveis, artefatos, ações, ferramentas entre outros, pode ser encontrado no APÊNDICE E.

### 4.4.3 **Papéis utilizados no processo de Teste de Software de Versão 2**

O processo de teste de software de versão 1 contava com 6 (seis) papéis de apoio ao processo e 1 (um) papel participante diretamente, porém, foi necessário adicionar mais 3 papéis participantes do processo para contemplar as melhorias ao processo de Teste de Software. O Quadro 4-5 mostra os papéis adicionados, suas descrições e a justificativa da adição.

Quadro 4-5 – Novos papéis adicionados no Processo de Teste de Software de Versão 2.

| **Papel / função** | **Descrição** | **Justificativa** |
|---|---|---|
| Líder de Teste | Responsável por especificar os padrões e metodologias que deverão ser utilizados pelos Analistas de Testes, bem como | A criação deste papel tornou-se necessária com o crescimento da equipe e com o aumento da complexidade da atividade, |

| Papel / função | Descrição | Justificativa |
|---|---|---|
| | melhorar o processo de teste, definir, melhorar e gerar templates; definir ferramentas de apoio; estabelecer prazos; elencar requisitos para os testes; e por fim definir as prioridades. | necessitando assim de um perfil dedicado à coordenação e evolução. |
| Analista de Homologação | Papel responsável por conduzir validações do Produto com o cliente, gerando o ambiente propício para homologação, bem como as estratégias a serem executadas ao longo dos trabalhos de validação do produto. | Este papel foi adicionado, pois o cliente solicitou um responsável para apresentar as funcionalidades do sistema, conduzir a atividade e que acompanhasse os Analistas de Homologação durante todo o processo de homologação. |
| Coordenador de Homologação | Responsável por especificar os padrões e metodologias que deverão ser utilizados pelos Analistas de Homologação, bem como melhorar o processo de Homologação, definir, melhorar e gerar *templates*; definir ferramentas de apoio; estabelecer prazos; elencar requisitos para homologação; | A criação deste papel tornou-se necessário com a necessidade de desenvolvimento de procedimentos e de um responsável pela atividade. |

### 4.4.4 **Artefatos utilizados no Processo de Teste de Software de Versão 2**

Nesta seção serão listados e descritos os artefatos adicionados e alterados no processo de teste de Software de Versão 2. O Quadro 4-6 lista, descreve e justifica a inclusão dos artefatos.

Quadro 4-6 – Artefatos adicionados e alterados no processo de Teste de Software Versão 2

| Artefato | Descrição | Justificativa |
|---|---|---|
| Projeto de Teste | Documento responsável por guardar casos de testes, os passos para reprodução, bem como dados de entrada e resultados esperados. | Para evitar redundância de dados e organizar de forma mais clara os casos de teste, o documento foi refatorado e teve seu objetivo focado apenas no projeto, não possuindo mais |

| Artefato | Descrição | Justificativa |
|---|---|---|
| | | informações sobre resultados. |
| Relatório de Teste | Documento responsável por guardar os resultados da execução dos projetos de testes, orientados por ciclo e rodadas. | A necessidade da criação deste documento foi de centralizar os resultados das execuções do projeto de teste a fim de gerar métricas e dos resultados serem consultados de maneira mais fáceis. |
| Registro de Ocorrências | Registro de ocorrências de defeitos, melhorias e mudanças. Registros são manipulados através de uma ferramenta de *bug tracking* | O registro de ocorrências nasceu da necessidade de haver um gerenciamento das defeitos encontradas na execução do processo de teste de software |
| Plano de Teste | Documentação que contém a descrição do processo de Teste de Software, fases, artefatos gerados, papéis e responsabilidades, tipos de testes a serem executados, tipo de ferramentas utilizadas. | Este documento nasceu da necessidade de ter um documento onde fosse apresentado e descrito o processo de teste de uma maneira geral onde, contendo procedimentos, papéis, responsabilidades e ilustração do processo. |
| Guia Operacional de Testes | Documento que contém as diretrizes para execução das técnicas e tipos de teste de software. | Surgiu da necessidade de um documento técnico para consulta de procedimentos, guiando os Analista de Testes nos procedimentos da disciplina adotados no projeto |

## 4.5 Análise do Processo de Teste de Software Versão 2

Esta seção mostra a Análise do Processo de Teste de Software de Versão 2 mostrando qual modelo de teste foi utilizado em sua definição e justificando a escolha, bem como, a comparação com outros processos de teste de softwares.

### 4.5.1 **Modelo do Processo de Teste de Software Versão 2**

O modelo utilizado como base para a construção do processo de teste foi o Modelo de Teste de Software em Espiral, pelo fato de ser compatível com o processo de desenvolvimento de software adotado na organização e, também, compatível com as melhorias adotadas no processo de teste de software.

Além das compatibilidades, o modelo espiral trás algumas vantagens que justifica sua escolha: a utilização de iterações, pelo fato de trabalhar com conceito de ciclos e rodadas, o que torna a estimativa de tempo de execução do ciclo de teste mais precisa; e a rápida capacidade de adaptação a mudanças e/ou novas funcionalidades.

Como característica específica do projeto, um pacote de entrega possui como condição a ser entregue que contenha, uma funcionalidade desenvolvida que gere ganhos negociais ao cliente e, uma entrega geralmente é complementar a outra entrega que já esteja em produção de maneira a somatizar as funcionalidades já existentes com novas funcionalidades que estão sendo entregues.

Com essa característica de entregas complementares, o modelo em Espiral foi o modelo que melhor se adequou ao modelo de processo utilizado no desenvolvimento. O processo de desenvolvimento, a cada novo ciclo, uma build candidata contendo novas funcionalidades e correções de defeitos é gerada e enviada a equipe de testes, que testa e devolve com os defeitos ou gera um documento atestando que o software está de acordo com o SLA.

As atividades e documentação do processo de teste de software tiveram como base o programa de melhoria do processo de software MPS.BR (Melhoria de Processo de Software Brasileiro) nas áreas de Verificação e Validação e o modelo TMM (Test Maturity Model). O Processo de Teste de Software de Versão 2, foi modelado com o objetivo de alcançar como aderência o Nível 3 do TMM, chamado de nível de integração, onde os testes são integrados ao ciclo de vida do software. Após o processo ser definido, foi executado uma vez, a fim de ser realizada uma análise de aderência com o TMM Nível 3, e as os ativos constantes nos processos de Verificação e Validação do MPB.BR.

### 4.5.2 **Análise de Trabalhos Correlatos**

Esta seção possui como objetivo a comparação do Processo de Teste de Software de Versão 2 com os processos de testes estudados como possíveis soluções para integrar ao processo de desenvolvimento da organização como processo de teste adotado. Entre vários trabalhos correlatos de modelagem de processo de teste de software, foram levantados e analisados suas aderências quanto: ao processo de desenvolvimento; o negócio do projeto em questão; e aderência ao MPS.BR.

Entre os mais relevantes, foram analisados cinco outros processos de testes que possuíam sua descrição disponível para consulta, adaptação e utilização, foram eles: o 3P x

3E (Filho, 2002); a Norma IEEE829 (IEEE, 1998); o processo da IBM AGEDIS; o processo desenvolvido pelo DATASUS; o processo desenvolvido pelo CenPRA (CenPRA, 2001); e o processo proposto nesse trabalho de Teste de Software Versão 2.

O Quadro 4-7 realiza a comparação das atividades entre os processos de testes levantados com o processo de teste pós melhorias desenvolvido, alem de realizar a aderência as fazes do MPS.BR das subáreas de verificação e validação. Já o Quadro 4-8 realiza a comparação dos artefatos gerados entre os processos levantados com o processo desenvolvido após as melhorias e o MPS.BR.

Quadro 4-7 – Comparação entre as atividades de processos de testes modelados e as exigências do MPS.BR para as áreas de VER e VAL.

| **Fases** | | | | | | **MPS.BR** | |
|---|---|---|---|---|---|---|---|
| **3P x 3E** | **Norma IEEE 829** | **AGEDIS** | **Datasus** | **CenPRA** | **ProTeste** | **VER** | **VAL** |
| | | Criação do Modelo | | | | | |
| Procedimentos Iniciais | | Levantamento das Informações | Iniciação | | Levantamento das Informações | VER1 – A fase deve possuir um procedimento onde seja identificado os produtos de trabalhos a serem avaliados. | VAL1 - A fase deve possuir um procedimento onde seja identificado os produtos de trabalhos a serem avaliados. |
| Planejamento | | | Planejamento/ Controle | | Planejamento/Controle | VER2 – Deve possuir procedimentos de escolha de estratégias de verificação, Estabelecimento de Cronogramas, Responsáveis e Levantamento de material necessário. | VAL 2 - Deve possuir procedimentos de escolha de estratégias de validação, Estabelecimento de Cronogramas, Responsáveis e Levantamento de material necessário. |
| Preparação/ Especificação | Preparação do Teste | Geração dos casos de Teste | Projeto e preparação do Ambiente | Preparação | Especificação | VER3 – definição dos critérios para verificação dos produtos de trabalhos a serem verificados, procedimentos a serem aplicados e ainda um ambiente é preparado. | VAL3 - definição dos critérios para validação dos produtos de trabalhos a serem verificados, procedimentos a serem aplicados e ainda um ambiente é preparado. |
| | | Revisão | | | | | |
| Execução | Execução | Execução | Execução | Execução | Execução/Elaboração de Scripts de Testes | VER 4 – Execução das atividades de verificação afim de garantir que o | VAL 4 - Execução das atividades de validação afim |

| Fases | | | | | | MPS.BR | |
|---|---|---|---|---|---|---|---|
| **3P x 3E** | **Norma IEEE 829** | **AGEDIS** | **Datasus** | **CenPRA** | **ProTeste** | **VER** | **VAL** |
| | | | | | Regressivos | produto esteja pronto para o ambiente de produção. / VER2 | de garantir que o produto esteja pronto para o ambiente de produção. / VAL2 |
| | Registro dos Testes | | | Registro dos Resultados | Registro de Resultados | VER 5 - Fase de identificação problemas são identificados e registrados. | VAL 5 - Fase de identificação problemas são identificados e registrados. |
| Entrega | | Análise | Encerramento | | Análise e Encerramento | VER 6 – Análise dos resultados e disponibilização em meio acessível a interessados. | VAL 6 – Análise dos resultados e disponibilização em meio acessível a interessados. VAL 7 – Fornecimento das evidências da realização de fases anteriores. |

Quadro 4-8 – Comparação entre os documentos de processos de testes modelados e as exigências do MPS.BR para as áreas de VER e VAL

| Artefatos | | | | | | MPS.BR | |
|---|---|---|---|---|---|---|---|
| **3P x 3E** | **Norma EEE 829** | **AGEDIS** | **DATASUS** | **Cenpra** | **ProTeste** | **VER** | **VAL** |
| | | Modelo Abstrato do software. | | | | | |
| Guia Operacional de Teste – GOT | | | | | Plano de Teste | VER1 – O documento deve possuir uma seção onde seja descrito os produtos de trabalhos a serem verificados. | VAL1 - O documento deve possuir uma seção onde seja descrito os produtos de trabalhos a serem validados. |
| | | | Registro de ocorrência | | | | |
| Estratégia de Teste | | Descrição dos critérios de Cobertura e Especificação do propósito. | | | Plano de Teste | VER2 – Deve possuir seções que evidencia a escolha de estratégias de verificação, Estabelecimento de Cronogramas, Responsáveis e Levantamento de material necessário. | VAL2 – Deve possuir seções que evidencia a escolha de estratégias de verificação, Estabelecimento de Cronogramas, Responsáveis e Levantamento de material necessário. |
| Análise de Riscos do Projeto de Testes | | | | | | VER 2/ VAL2 | |
| Plano de Teste | Plano do Teste | Geração automática dos casos de | Plano de Teste | Plano do Teste | | VER2 | VER2 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | teste. | | | | | |
| Registro e Controle das Diversas Versões do Produto | Especificação do Projeto de Teste | | | Especificação do Projeto de Teste | Projeto de Teste | VER3 – Deve obter uma seção que Defina os critérios para verificação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. | VAL3 – Deve obter uma seção que Defina os critérios para validação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. |
| Casos de Testes | Especificação de Casos de Teste | | Casos de Teste | Especificação de Casos de Teste | | VER3 – Deve obter uma seção que Defina os critérios para verificação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. | VAL3 – Deve obter uma seção que Defina os critérios para validação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. |
| Especificação das necessidades de dados de testes.<br><br>*Scripts* de Teste<br><br>Roteiros de Teste | Especificação de Procedimento de Teste | | | Especificação de Procedimento de Teste | | VER3 – Deve obter uma seção que Defina os critérios para verificação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. | VAL3 – Deve obter uma seção que Defina os critérios para validação dos produtos de trabalhos a serem avaliados, procedimentos a serem aplicados e ainda um ambiente é preparado. |
| *Scripts* de Teste | Diário de Teste | | Relatório de progresso de projeto | Diário de Teste | Relatório de Teste / Registro de Ocorrências | VER4 – Deve obter uma seção contendo evidencias verificação afim de garantir que o produto esteja pronto para o ambiente de produção. | VAL4 – Deve obter uma seção contendo evidencias validação afim de garantir que o produto esteja pronto para o ambiente de produção. |
| | | | Métricas do projeto de teste. | | | VER 4 – Execução das atividades de verificação afim de garantir que o produto esteja pronto para o ambiente de produção. | VAL 4- – Execução das atividades de validação afim de garantir que o produto esteja pronto para o ambiente de produção. |
| Relatório dos testes | Relatório de Incidente de Teste | Relatório de resultados e defeitos. | Relatório de resultados dos testes | Relatório de Incidente de Teste | Relatório de Teste / Registro de Ocorrências | VER 5 - Seção evidenciando que não conformidades são identificados e | VAL5 - Seção evidenciando que não conformidades são identificados e |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Relatório dos testes | Relatório de Incidente de Teste | Relatório de resultados e defeitos. | Relatório de resultados dos testes | Relatório de Incidente de Teste | Relatório de Teste / Registro de Ocorrências | VER 5 - Seção evidenciando que não conformidades são identificados e registrados. | VAL5 - Seção evidenciando que não conformidades são identificados e registrados. |
| Relatório Final dos testes | Relatório Resumo de Teste | | | | Publicação no Sistema de Controle de Versão | VER 6 – Documento ou seção contendo resultados e disponibilização em meio acessível a interessados. | VAL 6 – Documento ou seção contendo resultados e disponibilização em meio acessível a interessados. |

O processo de Teste de Software Versão 2 quando comparado ao processo de Teste 3P x 3E possui 3 pontos fortes. O primeiro diz respeito as etapas de Revisar Estratégias de Teste e Revisar Plano de Teste que ao invés de serem executadas no início do processo, são executadas ao final, juntamente a reunião de retrospectiva, afim de o processo evoluir de maneira mais rápida e com embasamento nas necessidades levantadas. O segundo ponto forte é que a adequação ao Plano de Gerência de Configuração, é alinhada com o processo de teste desde a fase de planejamento, uma vez, sendo a gerência de configuração tratar tanto a aplicação quando os documentos gerados pelo processo de teste, ao contrario do que o 3P x 3E faz, que é cuidar apenas da aplicação. O terceira ponto forte é que na preparação dos dados é feita juntamente com o projeto dos casos de uso, conforme as necessidades são relevadas, ao invés de serem preparados somente em tempo de execução.

Quando o processo de Teste Versão 2 é comparado ao processo de Teste IEEE 829 o processo de teste versão 2 possui 2 (dois) pontos Fortes. No primeiro o processo após as melhorias é um processo menos denso e custoso para a organização, tanto no quesito tempo quanto no quesito recursos, o forte controle e documentação do processo IEEE 829, o torna bastante custoso. Já no segundo ponto o processo após as melhorias exibe uma flexibilidade onde pode ser adequado tanto para um processo mais tradicional quando para um processo mais ágil, coisa que para o processo sugerido pela norma IEEE 829 torna-se bastante inviável.

Já quando comparamos o processo de Teste Versão 2, ao processo de Teste IEEE 829 o processo de Teste Versão 2 possui como ponto fraco, o fato do processo IEEE 829 pelo fato de possuir uma documentação completa e densa, faz com que o processo torne-se menos dependente de pessoas, ou seja, qualquer profissional com uma qualificação e conhecimento mínimos consegue dar vazão as atividades, uma vez, sempre bem documentadas.

O processo de Teste Versão 2 quando comparado ao processo de Teste AGEDIS possui dois pontos Fortes. No primeiro o processo após as melhorias possui como ponto forte

quando comparado ao processo AGEDIS sua robustez quanto a adequação dos casos de teste a novos requisitos. Já no segundo o processo AGEDIS é um processo que por ser bastante automatizado tem dificuldades em tratar mudanças no meio de um ciclo de teste.

Já quando o processo de Teste Versão 2 é comparado ao processo de Teste AGEDIS possui como ponto Fraco o fato do processo de Teste Versão 2, impacta de maneira considerável o processo de desenvolvimento de software, precisando de esforços para correções a mudanças durante e após o ciclo, enquanto que o processo AGEDIS pelo fato de ser simples e leve não causa grandes impactos no processo de desenvolvimento.

O processo de Teste Versão 2, quando comparado ao processo de Teste DATASUS possui como ponto forte o fato do processo do DATASUS possuir uma fase dedicada à geração de métricas, com isso, o processo de teste além de apontar falhas no produto gerado contribui com o desenvolvimento dos demais processos. Já o processo de teste de Software Verão 2 não possui uma atividade voltada a geração de métricas como principal função.

Já como pontos fracos, quando comparado o processo de Teste Versão 2 com o processo de Teste DATASUS possui dois pontos Fracos. O primeiro é que a fase de controle deveria ser transversal a todo o processo de maneira que as reuniões, deveriam ser feitas antes durante e depois de cada ciclo de processo e não apenas ao final dos testes. O segundo ponto fraco é que a ocorrência de falhas é registrada em dois documentos no Relatórios e o plano de execução, deixando assim informações redundantes, as falhas deveriam ser registradas em apenas um documento ou em uma ferramenta de *bugtracking*. No processo de Teste de Software Versão 2, as ocorrências são referenciadas em um documento e descritas na ferramenta de *bugtracking*.

Quando comparado o processo de Teste Versão 2, ao processo de Teste CenPra verificamos como ponto Forte a documentação de plano de teste que contempla as estratégias, processos e papeis de teste assim sendo instanciado a cada novo projeto, e acaba mesclando vários documentos descritos na norma IEEE 829 em um, deixando assim o processo menos oneroso.

Já como ponto fraco este processo não possui uma etapa onde sejam realizadas automação de teste, e testes regressivos. Os testes regressivos e/ou automatizados possuem importância no ganho de tempo em trabalhos repetitivos, desta maneira fazendo com que os analistas possam estar ao invés de realizando trabalhos repetitivos passar a realizar novas tarefas.

## 4.6 Considerações Finais

Como apresentado ao longo do capitulo, o projeto de Gestão de Fomento é um projeto grande e complexo, e por se tratar de um sistema de crédito bancário, é indispensável a qualidade no sistema e a alta disponibilidade do mesmo, sendo este produto o principal negócio do cliente, e portanto diretamente relacionado ao seu sucesso. Portanto, tratar o processo de teste de software para que se ganhe qualidade no produto é uma das premissas para que o projeto seja bem sucedido.

Com o processo de teste versão 1 apresentado, o fornecedor obteve problemas em fase de homologação do produto, obtendo altos retornos de defeitos e solicitação de mudanças, impedindo a homologação do mesmo e gerando conflitos entre Fornecedor e Cliente.

Portanto a utilização de um processo aderente ao processo de desenvolvimento e voltado para satisfação do SLA acordado entre as partes era premissa para o sucesso do projeto, porém, ao pesquisar os modelos já existentes e processos desenvolvidos no mercado, nenhum supriu totalmente as necessidades do projeto em função de particularidades do sistema, do cliente, do fornecedor e do processo de desenvolvimento, criando assim a necessidade do desenvolvimento de um processo aderente a essas necessidades e que cumprisse as normas de qualidade do MPS.BR e TMM.

Apos a criação deste processo o mesmo foi executado em duas entregas e seus resultados analisados e discutidos ao longo do Capítulo 5 a seguir.

# Capítulo 5 - **Avaliação do Processo de Teste de Software**

Este Capítulo possui como objetivo relatar a avaliação do Processo de Teste de Software de Versão 2 apresentado no Capitulo 4. Além do relato de sua execução, será descrito também na seção 5.1 a avaliação da execução desse processo através de resultados qualitativos e quantitativos da execução. Primeiramente o capítulo realiza uma avaliação da proposta, onde serão apresentadas as medidas para avaliação, as definições da avaliação e a aplicação das medidas. Na seção 5.2 será relatado o Estudo de Caso utilizado para a avaliação do processo de Teste de Software pós melhorias onde será apresentada uma introdução ao Estudo de caso, o contexto do estudo (aplicação), a aplicação prática do modelo do processo a análise quantitativa, análise qualitativa e considerações finais.

## 5.1 Metodologia de Avaliação

Segundo (Molinari, 2009), a busca pela melhor qualidade dos produtos e satisfação do cliente vem fazendo com que organizações que possuem sua atividade voltada para o desenvolvimento de software almejem melhorias de seus processos de desenvolvimento de software a fim de aumentar a qualidade do seu produto, acarretando em cada vez mais investimento em teste e qualidade de software. Porém, chegar a um processo de teste de software que satisfaça as necessidades da organização e do cliente, nem sempre é uma tarefa trivial. Vários processos passam por testes e aprimoramentos até se tornarem viáveis e aptos a suprir as necessidades acordadas entre a organização e o cliente, a partir do SLA.

Avaliações, acompanhamentos, testes e aprimoramentos nos processos geralmente possuem um custo elevado, entretanto, se não for medido e avaliado, grandes prejuízos para a organização e cliente podem ser gerados. Portanto, neste trabalho, para realizar uma avaliação do processo de teste de software de versão 2 foi necessário utilizar uma abordagem de medição cuja finalidade é identificar riscos, gargalos, avaliar mudanças, avaliar qualidade de artefatos e produtos gerados.

Para avaliação do processo foram adotados dois tipos de avaliações: uma qualitativa e outra quantitativa. A avaliação qualitativa ocorreu através da aplicação de questionários aos

participantes do processo contendo duas seções: perguntas a respeito do perfil do entrevistado e perguntas a respeito do processo de Teste de Software de Versão 2. Segundo (BASILI, CALDIERA and ROMBACH n.d.), para a avaliação quantitativa foi escolhido o método GQM (*Goal Question Metric*), onde a partir de objetivos definidos pela organização, é possível chegar a resultados que auxiliam a medição da qualidade do que está sendo produzido.

Objetivos – *Goal* – representa o objetivo principal, onde a organização deseja chegar.

Questões – *Questions* – Cada objetivo possui questões que uma vez respondidas pode-se avaliar se o objetivo foi alcançado.

Métricas – *Metrics* – Cada questão possui uma ou mais métricas. Com a geração das métricas, as questões podem ser respondidas com base.

O GQM é uma abordagem não sensível a escopo de projeto, tipo de processo ou produto e atua desde sua caracterização até o controle e aperfeiçoamento, para isso provê um framework que envolve três passos:

1. Elencar os principais objetivos do processo de medição;
2. Criar perguntas que respondam aos objetivos, para determinar se os objetivos foram atingidos;
3. Decidir o que necessita ser medido para que as perguntas sejam respondidas adequadamente.

O GQM possui objetivos que são transformados em perguntas, onde essas por sua vez possuem suas respostas embasadas em métricas. Após as métricas coletadas, essas irão compor as respostas das questões, que serão avaliadas para verificar o atendimento ou não aos objetivos.

O primeiro passo é definir objetivos a serem alcançadas no programa de medição, após a identificação das metas, um plano GQM deve ser elaborado para cada objetivo selecionado. O plano deve ser constituído de objetivos, que possuem um conjunto de questões quantificáveis que especificam as medidas adequadas para sua avaliação. As questões identificam informações necessárias para atingir os objetivos e as medidas definem operacionalmente os dados a serem coletados para responder as questões.

As metas GQM devem ser formuladas da seguinte forma: “Analisar o objetivo de estudo com a finalidade de objetivo com respeito ao enfoque do ponto de vista no seguinte contexto”. Sendo que os atributos destacados são definidos como:

- **Objetivo de estudo:** identifica o que será analisado. Exemplo: processo de software, projeto, documento, sistema, etc;
- **Objetivo:** por que o objeto será analisado. Exemplo: Avaliar, melhorar, monitorar, controlar, etc;
- **Enfoque:** identifica o atributo que será analisado. Exemplo: Confiabilidade, custos, correção, quantidade de falhas, etc;
- **Ponto de vista:** Quem irá utilizar os dados coletados? Exemplo: Gerente de Projeto, desenvolvedor, analista de teste, usuário, etc.

Como forma de aplicar o conhecimento discutido sobre a metodologia, as seções seguintes permitem uma analise das medições propostas para atender o cenário da fábrica de software relatado no capítulo 4.

## 5.2 Avaliação Quantitativa

Esta seção descreve como decorreu a avaliação quantitativa no processo de Teste de Software Versão 2, mostrando primeiramente na seção 5.2.1, sua definição e seu procedimento de coleta e na seção 5.2.2, discutindo seus resultados.

### 5.2.1 **Definição**

Para realizar a avaliação quantitativa do processo de teste de software após as melhorias, foi necessário definir os objetivos da organização, juntamente com a gerência para ter conhecimento onde a organização deseja chegar. O passo seguinte foi a elaboração das questões e posteriormente das métricas. O Quadro 5-1 mostra as metas definidas para o GQM para a avaliação quantitativa.

Quadro 5-1 - Meta definida para o GQM

| **Atributo** | **Valor para o Experimento** |
|---|---|
| Analisar (Objeto do Estudo) | Atividades, procedimentos e resultados realizados/obtidos pela execução do processo de de Teste de Software Proposto. |
| Com o propósito de (Objetivo) | Avaliar, aperfeiçoar e comparar |
| Com respeito a(o). | a Disciplina e Boas Práticas de Teste de Software.<br>As recomendações propostas no Guia de Implementação do MPS.BR referente aos processos de Validação e Verificação.<br>As recomendações de maturidade propostas no modelo TMM. |

| Atributo | Valor para o Experimento |
|---|---|
| | Ao cumprimento dos objetivos estabelecidos pela gerencia e coordenação de teste de software. . |
| Do ponto de vista (Perspectiva) | Gerentes de Projetos, Líder de Teste de Software, Analistas de Testes e Testadores. |
| No contexto da | Organização e ambiente em que ocorre o estudo de caso realizado. |

O objetivo geral deste estudo de caso é verificar se os processos de teste de software versão 2 atenderam aos objetivos da organização e as expectativas do Cliente quanto ao quesito qualidade. Através de métricas coletadas nas execuções do processo, entrevistas com os envolvidos no processo e análise de resultados, foi possível verificar se o processo atendeu as expectativas e objetivos da organização.

A execução do processo de teste em um projeto real possui como objetivos gerais:

- **Objetivo 1**: Disponibilizar critérios quantitativos para avaliar se as melhorias aplicadas estão sendo efetivas e contribuindo com a qualidade do produto;
- **Objetivo 2**: Garantir que o Produto gerado atenda ao SLA (*Service Level Agreement*) acordado entre Cliente e Fornecedor;
- **Objetivo 3**: Justificar o investimento realizado pela organização em teste de software;
- **Objetivo 4**: Validar as melhorias realizadas;
- **Objetivo 5**: Tornar o processo de teste de software aderente aos processos de melhorias do MPS.BR e TMM;
- **Objetivo 5**: Identificar pontos Fracos, Pontos fortes e novas oportunidades de melhorias;

Para este estudo de caso foram utilizadas duas organizações: o Cliente, um Banco público Federal; e o Fornecedor, uma Integradora de soluções em T.I (Tecnologia de Informação) de capital misto (público e privado) descritas com detalhes no Capítulo 4.

O cliente possui os seguintes setores que participam do projeto: a alta gerência; os gestores; os responsáveis pelo negócio atendido pelo sistema; e a área de T.I que são os responsáveis técnicos que realizam o acompanhamento do desenvolvimento do projeto além de serem futuros mantenedores do Sistema.

Em reunião entre a Gerencia do Projeto, o grupo de processo SEPG e o coordenador da equipe de teste de software, todos pertencentes a fornecedora, foram definidos 6 objetivos que o processo deveria atender após as melhorias implementadas, são eles:

- **Objetivo 1**: Disponibilizar para o cliente Sistemas com maior Qualidade;
- **Objetivo 2**: Completude na cobertura de teste;
- **Objetivo 3**: Abrangência ao modelo de Referencia MPS.BR para as áreas de trabalho de VER e VAL;
- **Objetivo 4**: Definir se os tipos de testes projetados para serem executados cobrem as regras de negócio e os requisitos contemplados no projeto;
- **Objetivo 5**: Definir o Custo benefício da equipe de teste para organização;
- **Objetivo 6**: Avaliar a aderência ao modelo de referencia TMM;

O Quadro 5-2 mostra as questões que foram elaboradas para cada um dos objetivos específicos da organização e as métricas que ajudam as questões a serem respondidas. Estas questões foram criadas através de reuniões entre a gerencia, SEPG e coordenação de Teste de Software da empresa. Para maior detalhes sobre as métricas consultar o Apêndice B.

Quadro 5-2 – Questões e métricas para serem utilizadas com GQM

| **Objetivo 1** | |
|---|---|
| **Questões** | **Métricas** |
| Q1: A porcentagem de defeitos críticos e importantes e triviais descobertos em homologação é menor do que os descobertos em fase de teste? | M1: A comparação da porcentagem de defeitos críticos, importantes e triviais encontrados em fase de homologação e testes no processo pós melhorias. |

| **Objetivo 2** | |
|---|---|
| **Questões** | **Métricas** |
| Q2: Consegue exaurir todos os testes relacionados aquela regra ou aquela U.C (Use Case) que está em teste? | M2: Comparação dos resultados dos projetos de testes descritos com os alvos de testes (Código fonte, Especificação de requisitos, Dicionário de Dados, integração). |
| Q3: Existem Atividades que deveriam ser executadas, e em função da sua não execução estão comprometendo a qualidade do Produto? | M3: Quantidade de Atividades e Boas práticas contempladas pelo processo de teste pós melhorias e que não estão sendo executadas e com a não execução estão comprometendo, levando em consideração ciclos de testes totalmente concluídos e bem sucedidos de acordo com o estabelecido no processo (Contemple todos os tipos de testes elencados |

| | |
|---|---|
| | para verificação e validação dos requisitos do produto). |

| **Objetivo 3** | |
|---|---|
| **Questões** | **Métricas** |
| Q4: As atividades do processo pós melhorias de teste de software estão com níveis de abrangência adequados em relação as recomendações pospostas pelo Guia de Implementação do MPS.BR no contexto de Verificação? | M4: Quantificar a adequação das evidências geradas na execução do processo de teste de software pós melhorias em relação às recomendações propostas no Guia de Implementação do modelo de referencia MPS.BR no contexto de Verificação. |

| **Objetivo 4** | |
|---|---|
| **Questões** | **Métricas** |
| Q5: Existem atividades e/ou procedimentos de testes que são exigidas para atendimento dos requisitos e regras de negócios e não estão sendo desenvolvidos? | M5: Quantidade de Atividades e/ou procedimentos de testes que são exigidos pela regra de negócio e não estão sendo contempladas durante a execução do processo pós melhorias pela equipe de testes. |

| **Objetivo 5** | |
|---|---|
| **Questões** | **Métricas** |
| Q6: Qual o custo/benefício da equipe de testes para a organização? | M6: Aplicando a lei 10 de Myers quantificar se vale a pena a organização continuar investindo em uma equipe de teste ou tratar os erros apenas quando forem reportados? |

| **Objetivo 6** | |
|---|---|
| **Questões** | **Métricas** |
| Q7: O processo de Teste de Software é aderente ao modelo de maturidade TMM? | M7: Grau de aderência da maturidade do processo em relação as boas práticas definidas no modelo TMM. |

Para a avaliação do processo gerado após as melhorias, foram realizadas duas etapas de coleta de resultados, uma qualitativa e outra quantitativa. A etapa qualitativa obteve seus dados coletados através de entrevistas com mais de 80% dos participantes executores do processo de teste. Já a etapa quantitativa obteve seus dados gerados a partir de auditoria a ativos do processo, entrevista e coleta de resultados.

Para o estudo de caso 37 pessoas participaram, onde destas, eram consultados dois Gestores por parte do cliente, duas pessoas desempenhavam papel de Gerente de Negócios,

dez Analistas de Requisitos, dois arquitetos de software, um Administrador de Dados, doze desenvolvedores e oito analistas de testes e três gerentes de Projetos.

As mudanças no processo de teste de software foram possíveis em função de o autor desta pesquisa ocupar a função de Líder de Testes bem como, membro do SEPG [8]e ter a flexibilidade de melhorar o processo de teste de software.

Para a avaliação do processo foram escolhidas duas entregas, Entrega 1 (E1) e Entrega 2 (E2), onde cada uma das entregas eram responsáveis por levar módulos do sistema primeiramente para homologação do cliente e posteriormente para produção. A Entrega 1 continha 9 Requisitos, onde desses 3 eram considerados complexos, 2 eram de complexidade média e 4 eram de complexidade simples. Na E2 foram entregues 6 Requisitos onde desses 4 eram complexos e 2 eram de complexidade media.

O tempo total do experimento foi de 3 meses onde a E1 aconteceu durante 1 mês e 10 dias e a E2 ocorreu durante 1 mês e 20 dias. Onde foram coletadas as métricas para posteriormente serem analisadas afim de responder se os objetivos estabelecidos para o processo de teste foram alcançados. A seguir serão descritas as Análises quantitativa e Qualitativa feitas no processo durante o período do experimento.

A avaliação quantitativa, ocorreu através de auditorias realizadas no repositório de ativos da organização, onde é possível ter acesso ao histórico de alterações e o responsável pelas alterações. A coleta de Dados ocorreu em um período de 06 de outubro de 2010 à 12 de fevereiro de 2011, como forma de verificar as evoluções do processo de Teste de Software Versão 2.

### 5.2.2 **Análise de Resultados**

A análise descreverá em termos práticos como foram realizadas as coletas das métricas, os resultados obtidos e posteriormente a análise dos resultados para cada um dos 6 (seis) objetivos definidos na seção 5.2. Para maior detalhamento das métricas utilizadas na análise quantitativa vide APÊNCICE A.

**Objetivo 1:** Disponibilizar para o Cliente sistemas com maior qualidade.

---

[8] SEPG (*Software Engineering Process Group*): Grupo responsável pelo melhoramento contínuo do processo de desenvolvimento de software organizacional, através de reuniões semanais periódicas.

Foram escolhidas duas entregas, Entrega 1 e Entrega 2 para a coleta das métricas. A Entrega 1 escolhida para o estudo de caso possuiu 9 Requisitos, onde destes 4 são considerados de complexidade simples, 2 de complexidade média e 3 são considerados complexos. Já a Entrega 2 eram no total 6 casos de uso, onde, destes 4 são considerados complexos e 2 possuem complexidade média.

A Quadro 5-3 contém informações sobre a métrica M1 utilizada para analisar o atendimento ao objetivo 1,onde, contém informações do que é a métrica, quando ocorreu, como foi realizada a coleta, onde foi realizada, o por que da métrica e o resultado esperado da mesma.

Quadro 5-3 – M1: Comparação da porcentagem de defeitos críticos, importantes e triviais.

| | |
|---|---|
| **O que?** | Comparar a quantidade de defeitos encontrados em fase de testes com a quantidade de defeitos encontrados em fase de homologação. |
| **Quando?** | Ao final do processo de Teste. |
| **Como?** | A coleta é realizada através de coleta dos defeitos registrado nos relatórios de teste e defeitos registrados na ferramenta de *Bug Tracking*. |
| **Onde?** | Na Fábrica de Software |
| **Por que?** | Os gerentes e coordenadores precisam verificar se um filtro efetivo de defeitos está sendo aplicado pela equipe de teste e se o sistema que está sendo disponibilizado para homologação possui qualidade. |
| **Resultado esperado** | A quantidade de defeitos descoberto na fase de teste seja pelo menos 80% maior que em homologação. |

Na etapa de testes da Entrega 1, que ocorreu durante um mês e 10 dias, onde, na etapa de testes foram encontrados 238 defeitos, desses 46 foram considerados triviais, 170 foram considerados importantes enquanto que 22 foram considerados defeitos críticos verificados na Figura 5-1. Na etapa de homologação foram descobertos um total de 4 defeitos, onde desses os 4 foram considerados críticos como mostra a Figura 5-2.

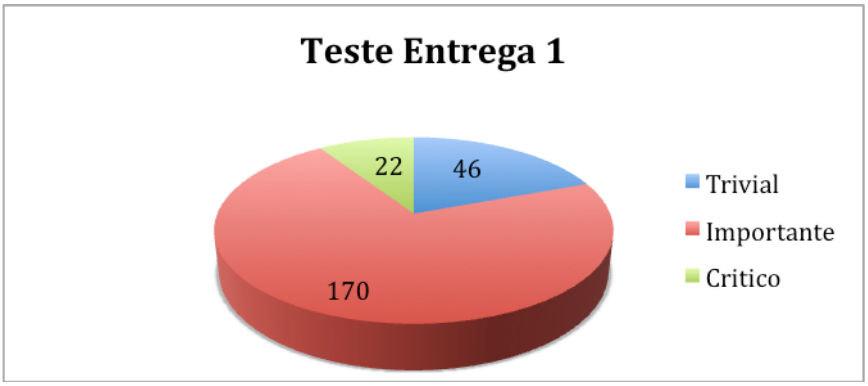


Figura 5-1 – Ocorrências de criticidade de defeitos em fase de Teste para E1.

Portanto, a fase de teste descobriu 98,34% dos defeitos conhecidos do sistema antes do mesmo ser colocado em homologação e posteriormente em produção, mostrados na Figura 5-3.

Porém, mesmo com uma porcentagem baixa de erros descobertos na fase de Homologação o produto não atingiu ao SLA estabelecido entre Cliente e Fornecedor pelo fato de em Homologação ter sito encontrado defeitos críticos o que independente de sua porcentagem ou quantidade inviabiliza a homologação do produto. Sendo assim para a Entrega 1 o Objetivo 1 falhou.


Figura 5-2 – Ocorrência de criticidade de defeitos em fase de Homologação para E1.

Ao final da etapa de testes da E1, foi promovido um *workshop* interno, com o objetivo de diminuir o risco de ocorrências de defeitos em fases que procedem a fase de teste.

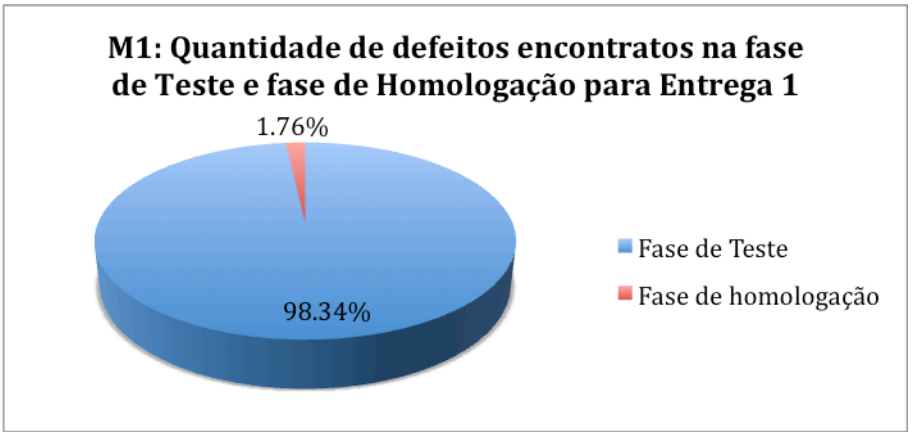


Figura 5-3 – Ocorrências de defeitos em fase de Teste x Homologação para E1.

Após o *workshop*, a etapa de testes da Entrega 2, que ocorreu durante um mês e vinte dias, foram encontrados 464 defeitos em fase de teste, onde desses 101 foram considerados

triviais, 316 foram considerados importantes enquanto que 47 foram considerados defeitos críticos podendo ser verificado na Figura 5-4.

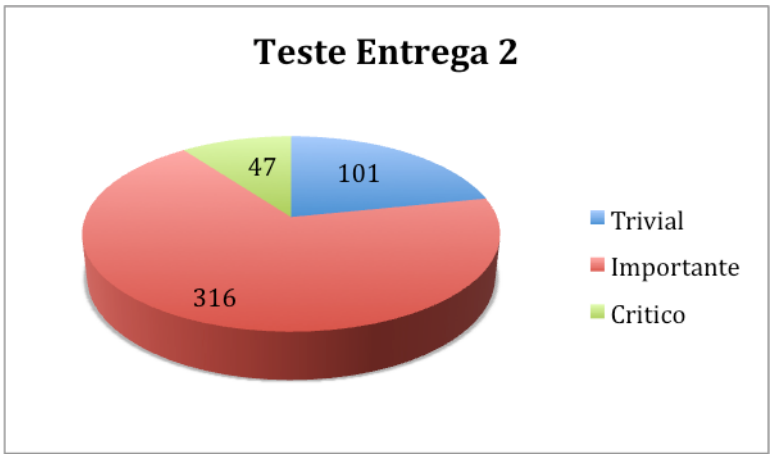


Figura 5-4 – Ocorrências de criticidade de defeitos em fase de Teste para E2.

Já na etapa de homologação foram descobertos um total de 12 defeitos, onde desses, 7 foram considerados triviais, 5 foram considerados importantes e não houveram ocorrências de defeitos críticos na homologação como mostra a Figura 5-5.

Portanto, a fase de teste descobriu 97,48% dos defeitos conhecidos do sistema antes do mesmo ser colocado em homologação atendendo as expectativas da gerencia de projetos e coordenador de Testes e também respondendo positivamente ao workshop de treinamento interno, como mostra a Figura 5-6.

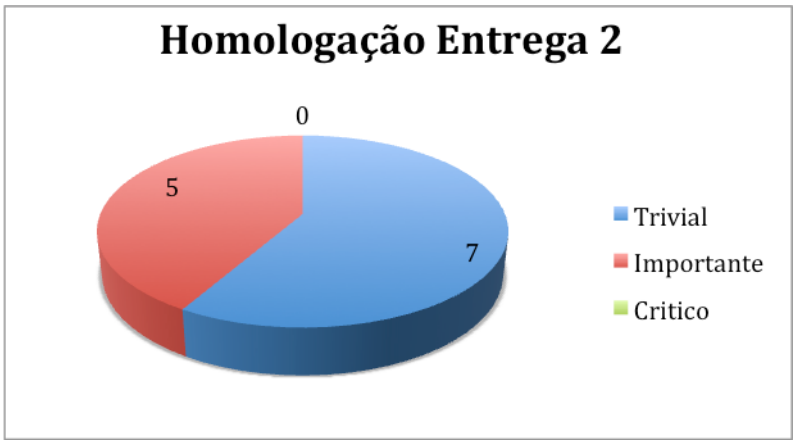


Figura 5-5 – Ocorrência de criticidade de defeitos em fase de Homologação para E2.

O motivo da Homologação da E2 obter maior sucesso no quesito SLA foi devido ao workshop, onde neste, foi realizado um treinamento do processo, dos procedimentos e trabalhado o comprometimento da mesma com os resultados, o que acabou refletindo um produto de maior qualidade e que atendeu ao SLA e consequentemente ao Objetivo 1.

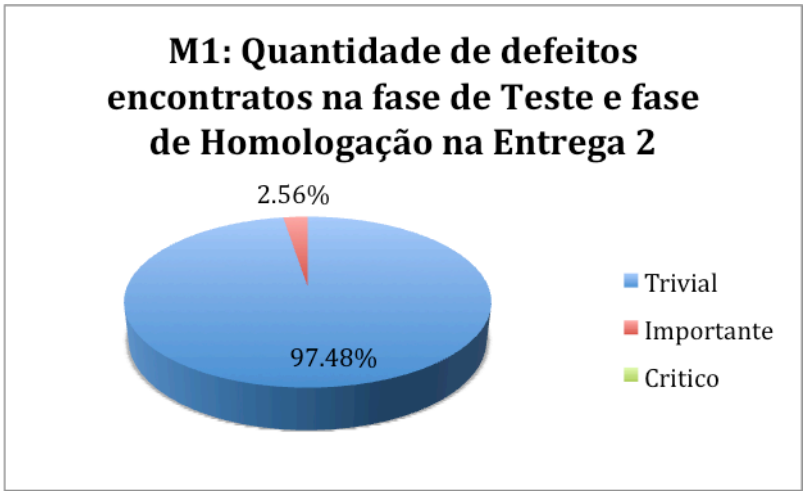


Figura 5-6 – Ocorrências de defeitos em fase de Teste x Homologação para E2.

**Objetivo 2:** Completitude da Cobertura de Testes.

O objetivo 2 é avaliado através de duas métricas M2 e M3, onde a M2 trata da comparação dos resultados de testes descritos com os alvos de teste, ou seja, mede se os testes estão de fato atendendo aos itens alvos listados no processo. A M3, trata da quantidade de Atividades e Boas práticas que não são contempladas na sua execução pelo processo pós melhorias, levando em consideração ciclos de testes totalmente concluídos e bem sucedidos de acordo com o estabelecido no processo. O Quadro 5-4 contém informações sobre a métrica M2 utilizada para analisar o atendimento ao objetivo 2.

Quadro 5-4 – M2: Comparação dos resultados de projetos de testes descritos com alvos de testes.

| | |
|---|---|
| **O que?** | Quantificar a cobertura real da equipe de teste quanto aos ativos do constantes do processo de testes que deveriam ser testados. |
| **Quando?** | Ao final da execução do processo de Teste. |
| **Como?** | A coleta foi realizada através de entrevistas com os analistas de testes e auditoria aos projetos de teste e plano de teste. |
| **Onde?** | Na Fábrica de Software ao final da execução do processo de teste de software. |
| **Por quê?** | Os gerentes e coordenadores necessitam se o processo de teste executado está de fato tendo completitude da cobertura ao sistema. |
| **Resultado esperado** | O resultado esperado é que todos os itens alvo de testes sejam contemplados nos projetos de testes. |

Nesta métrica, foi adotado o mesmo padrão de avaliação de atendimento das atividades que é adotado no MPS.BR, onde, são dadas 5 pontuações: T, L, P, N e NA. Onde a pontuação T significa que o processo atente Totalmente ao item exigido, L significa que o

item é atendido Largamente, P significa que o item é atendido Parcialmente o N significa que o item não é atendido e o NA significa que o item Não se aplica ao processo.

A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação, onde, através de entrevistas e auditorias em ativos do processo foi avaliado se a execução dos testes estavam de fato obtendo uma completude nos itens alvos de testes.

Como resultado, foi verificado que dos 8 itens alvo de teste listados no plano de teste, 2 itens receberam avaliação de Totalmente atendidos, 3 itens receberam avaliação de largamente atendidos e os últimos 3 itens receberam avaliação de parcialmente atendidos.

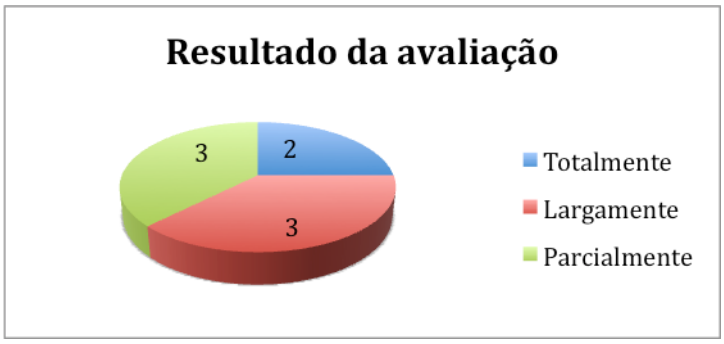


Figura 5-7 – Resultado da avaliação das atividades.

Os itens que receberam a avaliação de totalmente atendidos foram: Especificação de Requisitos e Regras de negócio, pelo modo que é feito, o projeto de teste possui sua orientação por Especificação de Requisitos e Regras de Negócio, desta maneira todos os projetos de teste cobrem totalmente os itens listados acima.

Já os três itens que tiveram avaliação como Largamente atendidos foram: Banco de Dados, Dicionário de Dados e Código Fonte. Os itens banco de dados, Dicionário de dados e Código Fonte receberam essa avaliação em função de o teste ser orientado a funcionalidade através de dados de entrada e respostas, desta maneira apenas os registros e campos que servem de entrada e de saída são testados diretamente, campos e funções que não possui interação diretamente com entradas e saídas de telas são testados indiretamente pelas funcionalidades.

Os itens que receberam avaliação de Parcialmente atendidos foram: Integração Externa, Padrão de Interface e Arquitetura do sistema.

O item integração externa obteve a avaliação de Parcialmente atendidos, pelo fato de os testes que envolvem integração serem feitos apenas na parte da integração que tange a funcionalidade do sistema e não a integração totalmente. Como exemplo um campo que é

extraído via *WebService* de um sistema externo e é utilizado no sistema alvo de teste como uma lista de seleção, possui seu teste apenas para saber se o Domínio das informações estão corretamente preenchidos e se satisfaz as condições de Regra de Negócio e Requisitos, não sendo alvo do teste diretamente a integração em si.

O teste de Arquitetura do Sistema, acontece de maneira parcial, conforme defeitos mais complexos e específicos são encontrados em meio as funcionalidades, a arquitetura do sistema não tem sido alvo principal de teste, que possui sua orientação por funcionalidade.

Para melhor atender a esse teste foi realizando um tutoriamento e um treinamento interno com a equipe para que através de ferramentas e procedimentos de teste caixa branca, pudesse passar a testar as integrações efetivamente.

Para o teste de Padrão de interface, foi solicitado uma documentação mais detalhada sobre o padrão contendo além dos comportamentos padrões de componentes, campos e máscaras, fosse detalhado também a escala de cores e tamanhos. Além dessas informações, novos tipos de testes terão que ser incluídos no projeto de teste. Como teste de interface, testes de banco de dados e testes de integração.

Para que o teste de Banco de Dados fosse mais efetivo foi solicitado a equipe de Administradores de Dados, um *Workshop* interno onde os Administradores de Dados explanariam sobre a modelagem em questão e as ferramentas de uso, para que testes mais específicos pudessem ser feitos.

O Quadro 5-5 descreve o que é a métrica M3, quando foi aplicada, como, onde, porque e resultados esperados. A M3 juntamente com a M2 irão compor os resultados para o objetivo 2.

Quadro 5-5 – M3: Quantidade de Atividades e Boas práticas contempladas pelo processo de teste V2.

| | |
|---|---|
| **O que?** | Verificar a quantidade de Atividades e boas práticas que não são executadas e com isso comprometendo a qualidade do produto. |
| **Quando?** | Ao final da execução do processo de Teste. |
| **Como?** | A coleta é realizada através de auditorias a ativos do processo e entrevistas a executores do processo onde as atividades serão classificadas como Totalmente executadas, Largamente executada, Parcialmente executada e Não executada. |
| **Onde?** | Na Fábrica de Software, ao final da execução do processo de teste. |
| **Por quê?** | Os gerentes e coordenadores necessitam verificar se todas as atividades descritas no processo estão sendo executadas. |
| **Resultado esperado** | Que todas as atividades do processo estejam sendo executadas. |

A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação. Onde através de entrevistas e auditorias em ativos do processo realizadas foram coletadas as métricas.

No total são 8 macro atividades descritas no processo cada uma contendo suas sub atividades e ações associadas. Das 8 atividades listadas 4 foram classificadas como Totalmente executadas, 3 foram classificada como largamente executada e 1 foram classificadas como parcialmente executadas como mostrado na Figura 5-8.

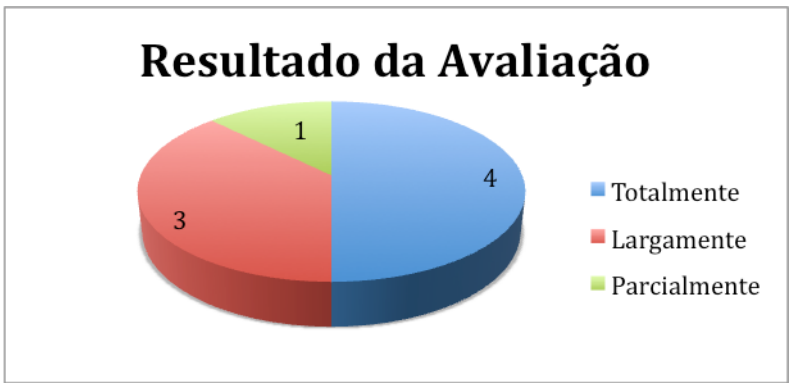


Figura 5-8 – Resultado da avaliação das atividades.

As atividades de Gerar/Rever Projeto de Teste, Registrar Defeitos, Gerar/Rever Scripts de Teste e Executar Projeto/Scripts de Teste receberam avaliação de totalmente atendidas.

Já as 3 atividades que receberam avaliação de largamente executadas foram: Gerar/Rever Guia Operacional de Teste, Gerar/Rever Plano de Teste, Gerar/Rever Plano de Homologação. Essas atividades receberam esta avaliação em função da maneira com que a ação de Rever Guia operacional, Rever Plano de Teste e Rever Plano de homologação são executadas. O fato dessas atividades receberam a avaliação de largamente executada foi em função da ação de "Rever" ser executada apenas quando solicitado, ao invés de ser executado sempre que mudanças e/ou replanejamentos são realizados.

A atividade de Liberar Build recebeu a avaliação de Parcialmente executado, em função de ter ocorrido em umas das entregas a liberação da build para homologação sem a aprovação e liberação da equipe de testes.

Portanto, o Objetivo 2 foi parcialmente contemplado, enquanto que sua expectativa era que todas as avaliações para atividades e Tipo de Testes fossem de largamente e/ou de Totalmente ou Largamente atendidos. Para melhorar a avaliação a etapa de Liberar Build, foi proposto que quando o prazo de entrega for esgotado e os testes ainda não tiverem sido

concluídos, a gerência tentará negociar com cliente um novo prazo, ou realizar uma entrega parcial apenas dos módulos que já estiverem sido testados por completo.

**Objetivo 3:** Abrangência ao Modelo de Referencia MPS.BR.

O objetivo 3 é formado por uma métrica que é composta de medidas adotadas no MPS.BR para avaliação de processos, que possui como objetivo para organização verificar se o processo de teste é aderente as recomendações do MPS.BR para os processos de VER e VAL. O Quadro 5-6 descreve a métrica adotada para auxílio da análise de satisfação do objetivo.

Quadro 5-6 – M4: Avaliar a complexidade das atividades.

| | |
|---|---|
| **O que?** | Quantificar a adequação ao modelo do MPS.BR no contexto de VER e VAL afim de verificar se o processo de teste é aderente ao exigido pelo MPS.BR. |
| **Quando?** | Ao final da execução do processo de Teste. |
| **Como?** | A coleta é realizada através de entrevistas com participantes do processo e através de auditorias em ativos do processo realizada por auditores cadastrados e atuantes da SOFTEX. |
| **Onde?** | Na Fábrica de Software |
| **Por quê?** | Os gerentes e coordenadores necessitam verificar a maturidade do processo de teste, bem como sua aderência ao modelo de Referencia, afim de submeter o mesmo futuramente a avaliação. |
| **Resultado esperado** | Que o processo de teste seja aderente ao modelo de referencia no contexto de VER e VAL do MPS.BR. |

A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação. Onde através de entrevistas e auditorias em ativos do processo realizadas por um avaliador credenciado na SOFTEX.

O MPS.BR adota um padrão de pontuação onde são dadas três pontuações: T, L, P, N e NA. Onde a pontuação T significa que o processo atente Totalmente ao item exigido, L significa que o item é atendido Largamente, P significa que o item é atendido Parcialmente o N significa que o item não é atendido e o NA significa que o item Não se aplica ao processo.

O resultado da avaliação para o processo de Validação dos sete itens exigidos pelo MPS.BR todos receberam T, ou seja, Totalmente aderente ao MPS.BR como mostra a Figura 5-9.

Para o processo de Verificação para os 6 itens constantes, 4 itens receberam avaliação T, enquanto 1 item recebeu avaliação L e 1 item também recebeu avaliação P como mostra a Figura 5-10.

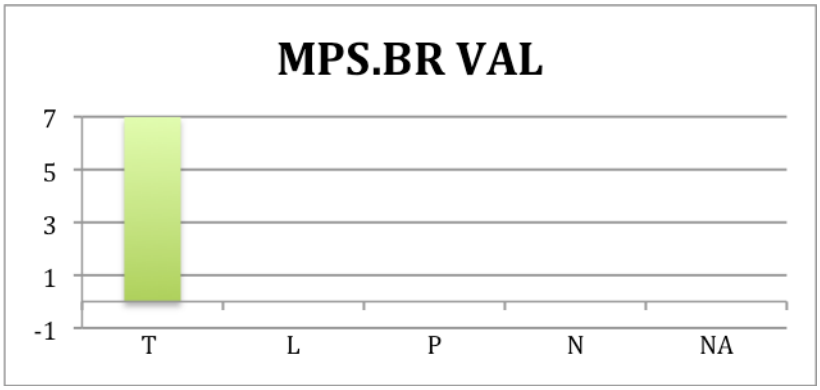


Figura 5-9 – Avaliação de itens para o processo de validação.

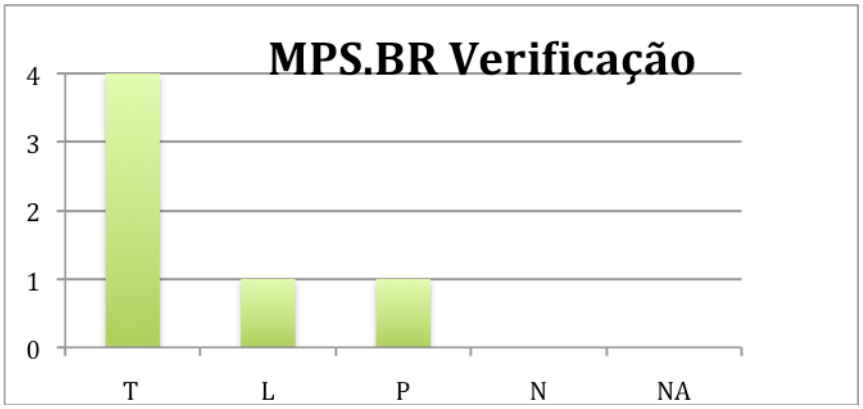


Figura 5-10 – Avaliação de itens para o processo de verificação.

Para os itens que receberam a avaliação de Largamente e Parcialmente, foram discutidos na reunião de retrospectiva e alguns ajustes foram realizados. Para o item VER 4 (Atividades de verificação, incluindo testes e revisões por pares, são executadas) , para a revisão por pares foram feitos escalas entre os analistas e institucionalizado a obrigatoriedade de cada U.C testado obrigatoriamente deve ser testado por pelo menos dois analistas diferentes.

Para o item VER 6 (Resultados de atividades de verificação são analisados e disponibilizados para as partes interessadas) que recebeu a avaliação Parcialmente atendido foi tratado colocando a atividade de envio de e-mail notificando o andamento da qualidade do sistema oficialmente no processo, uma vez a atividade ser executada mesmo não estando institucionalizada.

**Objetivo 4:** tipos de testes projetados contemplam as regras de negócio os requisitos?

O objetivo 4 consiste em definir se os tipos de testes projetados para serem executados cobrem as regras de negócio e os requisitos contemplados no projeto. Para avaliação de sua satisfação é utilizada como instrumento de análise a M5 descrita no Quadro 5-7.

Quadro 5-7 – M5: Atividades de testes que são exigidos pela regra de negócio e não contempladas pelo processo v2.

| | |
|---|---|
| **O que?** | Verificar se os tipos de testes projetados cobrem as exigências das regras de negócio e dos requisitos. |
| **Quando?** | Ao final da execução do processo de Teste. |
| **Como?** | A coleta é realizada através de entrevistas com participantes do processo e através de auditorias em ativos do processo realizada por auditores cadastrados e atuantes da SOFTEX. |
| **Onde?** | Na Fábrica de Software, ao final do processo de teste de software. |
| **Por quê?** | Os gerentes e coordenadores necessitam se os tipos de testes projetados são os suficientes para testar as regras de negócios e os requisitos. |
| **Resultado esperado** | Que os tipos de testes descritos cubram as exigências das regras de negócio e requisitos. |

A avaliação foi realizada através entrevistas e análise a ativos e chegou-se aos seguintes resultados.

Três tipos de testes deveriam ser executados para que as exigências de requisitos e Regras de negócios sejam melhor atendidas: o primeiro tipo de teste é o teste de arquitetura que obtém como item alvo de teste a arquitetura do sistema, a maneira indireta que esse tipo de teste é executado deixa lacunas importantes sem teste, de maneira a aumentar riscos de ocorrência de defeitos no sistema em etapas futuras. O segundo tipo, é o teste de instalação, que não é executado e em função da sua não execução alguns defeitos apareceram em função de instalações incompletas em ambientes diferentes.

O terceira tipo é o teste de ambiente propriamente digo, que não é executado em função de tempo, e a sua não execução, vários defeitos foram gerados por diferenças de ambientes e configurações de ambientes.

Com esses resultados, esse objetivo não foi atingido, porém foi gerado ações como modelagem dos tipos de testes levantados para que passem a integrar e fazer parte do processo de teste.

**Objetivo 5:** Definir o custo benefício da equipe de testes para a organização.

O objetivo 5 consiste em informar à gerência se é vantajoso continuar investindo e uma equipe de teste para a organização. Este objetivo, possui sua avaliação analisada por uma métrica que quantifica os custos financeiros de uma equipe e os defeitos descobertos em fase

de testes com os custos financeiros se não houvesse uma equipe de testes e os defeitos fossem encontrados em fases posteriores através de lei 10 de Myers. O Quadro 5-8 explica os objetivo da métrica.

Quadro 5-8 – M6: Quantificar o número de defeitos de acordo com sua complexidade

| | |
|---|---|
| **O que?** | Verificar se investir financeiramente em uma equipe de teste esta dando retorno ou é mais vantajoso esperar que os defeitos sejam descobertos em outras fases para depois serem corrigidos. |
| **Quando?** | Ao final da execução do processo de homologação. |
| **Como?** | Quantificação é realizada pós o fechamento da Homologação. |
| **Onde?** | Na Fábrica de Software, ao final da execução do processo de Homologação. |
| **Por quê?** | Os gerentes e coordenadores necessitam verificar se seus investimentos em qualidade estão trazendo retorno positivo a organização. |
| **Resultado esperado** | Que investir em uma equipe de teste seja mais vantajoso do que deixar com que os defeitos sejam descobertos pelos clientes. |

Ao final da execução do processo de homologação, foi realizado um levantamento dos defeitos descobertos na etapa de testes e dos defeitos descobertos pelo cliente na etapa de Homologação. Após o levantamento através de reunião com o coordenador de desenvolvimento e coordenador de Requisitos, foram levantados o tempo gasto de correção para cada categoria de defeitos. Foi feito o levantamento de custo de correção de defeitos para cada categoria (Trivial, importante e Critico).

Geralmente os defeitos triviais necessitam em media de duas horas de correção e utilizam apenas analistas de desenvolvimento em sua correção. Os defeitos importantes, geralmente utilizam em media três horas de um analista de desenvolvimento e uma hora trabalhada de um analista de Requisito. E por fim para os defeitos categorizados como críticos, em média necessitam de seis horas de um analista de desenvolvimento e duas horas de um analista de Requisitos.

Diante dessas médias chegou-se aos seguintes valores:

- O custo aproximado de correção para um defeito trivial é de R$ 39,77;
- O custo aproximado de correção para um defeito importante é de R$ 90,81;
- O custo aproximado de correção para um defeito crítico é de R$ 181,63;

Foi também parametrizado o tempo médio gasto pela equipe de teste para testar os Casos de Uso por complexidade onde para um Caso de uso de complexidade simples gasta-se em media 3h, já para um Caso de Uso de complexidade média, gasta-se 5h e para um de complexidade critica 12h. Sabendo que dos 10 U.C´s testados 5 U.C´s eram considerados simples enquanto que 2 eram de complexidade média e 3 eram complexos.

Com base no tempo gasto pela equipe de teste chegou-se as seguintes previsões de custos:

- O custo de teste para um U.C de complexidade simples é de R$ 59,66;
- Para de complexidade media é de R$ 99,43;
- E para de complexidade alta o custo médio foi de R$ 238,64;

Totalizando, o custo de teste total contendo 4 Ciclos de teste onde cada ciclo contém de 10 a 12 rodadas foi de R$ 10.480,11. Já o custo de correções dos defeitos encontrados na fase de teste foi de R$ 21.263,70

Segundo Emerson Rios (2007), sempre que um defeito for encontrado em determinada versão o sistema, após sua correção a versão deve ser ré submetida a testes regressivos para garantir que a correção aplicada na versão não gerou novos defeitos. No total foram dois pacotes de correções aplicadas durante o Período de teste, portando o sistema foi testado 3 vezes.

De posse de dados apresentados sobre custos, foi aplicado a lei 10 de Myers onde diz que prova para cada etapa do processo que o defeito persiste sem ser encontrado e/ou corrigido, seu custo aumenta em 10 vezes.

O custo total de Teste de software somado ao custo das correções dos defeitos encontrados foi de aproximadamente R$ 31.743,81. Aplicando a Lei 10 de Myers ao custo dos defeitos, se a mesma quantidade de defeitos fossem encontrados em fase de homologação e só então fossem corrigidos, o custo desses defeitos passariam a ser de aproximadamente R$ 212.637,00. E se esses mesmos defeitos fossem encontrados em produção o custo saltaria para R$ 2.126,370,03, como mostra a Figura 5-11.

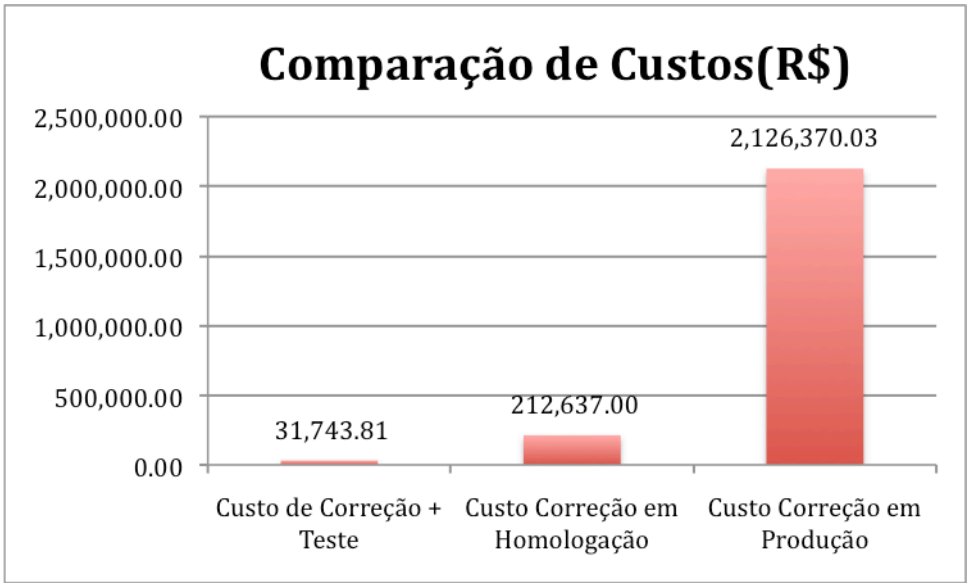


Figura 5-11 – Comparação de custos de defeitos.

Portanto, mesmo investindo em uma equipe de Teste a organização possui um decremento em seu custo de correção quando comparado a fase de homologação de R$ 180.893,19. Outro fator tão importante quando a economia de recursos financeiros é a imagem da organização perante ao cliente, que ao entregar software com mais qualidade o cliente tende a fechar novos projetos e a imagem da organização é fortalecida.

**Objetivo 6:** Avaliar a aderência dos ativos constantes no processo de teste ao modelo de maturidade TMM.

O Objetivo 6 avaliar a aderência dos ativos constantes no processo de teste ao modelo de maturidade TMM, consiste em a organização, além de pautar seu processo de teste em um programa de melhorias de processo geral, pautar seu processo também em um programa de melhoria específicos para teste. A métrica M7 possui sua descrição apresentada no Quadro 5-9.

Quadro 5-9 – M7: Analisar o grau de aderência das atividades ao modelo de maturidade TMM.

| | |
|---|---|
| **O que?** | Verificar em que nível se encontra o processo de teste de software quando comparado ao TMM. |
| **Quando?** | Ao final da execução do processo de Testes. |
| **Como?** | Através de inspeção em artefatos e entrevistas com os participantes do processo. |
| **Onde?** | Na Fábrica de Software, ao final da execução do processo de Teste de Software. |
| **Por quê?** | Os gerentes e coordenadores necessitam verificar se o processo pós melhorias estão de acordo com padrões específicos de Teste de Software. |
| **Resultado esperado** | Que o processo de Teste de software Esteja pelo menos no nível 3 do TMM. |

Ao final da execução do processo de teste de software, foi realizado um levantamento e avaliação das atividades e ativos do processo afim de realizar a aderência ao processo de melhoria TMM e com isso revelar o nível de maturidade do processo de teste de software da organização. A avaliação foi realizada por um avaliador autorizado da SOFTEX. O método de avaliação foi o mesmo utilizado para verificar a aderência ao MPS.BR descrito na métrica M4.

O nível revelado pela avaliação foi que o processo de teste estaria em nível de maturidade 3. Para cumprir as exigências no nível 1 o teste tem que ser feito pela equipe de desenvolvimento. As evidências são encontradas nos testes de unidade caixa branca gerados pela equipe de desenvolvimento.

O nível 2 por sua vez possui 3 exigências: a primeira é que seja desenvolvidos os objetivos do teste, e essa evidência foi identificada no Plano de Teste e no objetivo gerado com a definição do SLA; a segunda é que seja iniciado um processo de planejamento de teste, e essa exigência é evidenciada na descrição do processo de teste de software;  a terceira exigência é que as técnicas e métodos básicos de teste sejam institucionalizados, e é evidenciado quando o plano de Teste é disponível em um CVS (Control Version System), institucional.

O nível 3 possui 4 exigências onde a primeira é que seja estabelecida uma organização de teste de software e essa exigência foi evidenciada no documento de Plano de Teste e no Manual Operacional de Teste. A segunda exigência é que o processo de teste seja integrado no ciclo de vida do processo de desenvolvimento e é evidenciado no processo geral de desenvolvimento de software e é também observado no cronograma geral do projeto. A terceira exigência trata-se de controlar e monitorar o processo de software que é evidenciada através da planilha de acompanhamento de teste, cronograma e Relatório de Teste. E a Quarta exigência é que haja um programa de treinamento que no processo em questão é evidenciado através da realização de Workshop Interno e através do Manual Operacional de Teste.

Os resultados foram satisfatórios e atingiram os objetivos da gerência, porém será feito uma avaliação e estudo para que o processo evolua até o nível 4, que não obteve nenhuma de suas exigências atingidas, bem como o nível 5.

## 5.3 Avaliação Qualitativa

Como forma de complemento à avaliação quantitativa, foi importante realizar uma coleta também de forma qualitativa, para obter a opinião (*feedback*) dos membros da equipe de Teste de Software.

### 5.3.1 **Definição**

A metodologia utilizada para realizar a avaliação qualitativa foi através da aplicação de questionários. Para a criação do questionário foram necessárias reuniões entre o coordenador da equipe de teste de software e a gerência do projeto, com o objetivo de realizar o levantamento de outras necessidades a qual a avaliação quantitativa não abrangia.

Para melhor organização, o questionário foi dividido em 3 (três) seções: o perfil dos participantes; a experiência dos participantes; e a análise das atividades do processo de Teste de Software de Versão 2.

A primeira seção do questionário possui o objetivo definir o perfil dos participantes enquanto que a segunda seção do questionário, tem o objetivo de definir a experiência e conhecimento dos participantes quanto ao conhecimento em: teste de software, engenharia de software e qualidade de software.

A terceira seção do questionário apresenta uma forma de avaliação do processo de Teste de Software de Versão 2, onde, tem como finalidade avaliar cada atividade do processo e através desta avaliação verificar se o processo necessita de alguma melhoria de acordo com a visão dos entrevistados.

O questionário possuía três critérios de avaliação para cada item, são eles:

- **Presença**: se uma determinada atividade está presente e é executada no processo de Teste de Software de Versão 2;
- **Utilidade**: se uma determinada atividade tem utilidade no processo de Teste de Software de Versão 2;
- **Adequação**: se uma determinada atividade necessita de alguma adequação para o processo de Teste de Software de versão 2.

Para facilitar o entendimento dos resultados, o Quadro 5-10, mostra os critérios e suas possíveis ocorrências agrupadas.

Quadro 5-10 – Critérios para agrupamento do questionário

| Presença | Utilidade | Adequação do nível de detalhamento |
|---|---|---|
| 1. Não é oferecido pelo processo e não gostaria que tivesse disponível. | 1. Não é útil. | 1. O detalhamento deve ser aumentado. |
| 2. Não oferecido, mas gostaria que tivesse disponível. | 2. Provavelmente é útil, mas ainda não apliquei. | 2. O detalhamento não precisa ser modificado. |
| 3. Oferecido, parcialmente. | 3. É útil e já apliquei em diferentes implementações. | 3. O detalhamento deve ser diminuído. |
| 4. Oferecido. | | |

Após a definição dos critérios do questionário, foi realizada uma verificação com objetivo de encontrar erros. Esse questionário de teste foi distribuído para três revisores que haviam conhecimento em teste de software mas não participaram da atividade de teste de software na fábrica. Posteriormente, foi realizada a coleta dos questionário para avaliação dos pareceres e possíveis alterações ou correções.

Antes da distribuição do questionário para os 8 entrevistados, o que representam mais de 80% dos participantes do processo, os mesmos foram convidadas a participar de uma explicação, onde, através a apresentação do questionário, o objetivo, o termo de confidencialidade e a apresentação do Processo de Teste Versão 2, os entrevistados foram solicitadas a responder o questionário. O tempo máximo de resposta foi estipulado de 25 minutos.

Todos os questionários foram recolhidos e seus dados foram consolidados em uma planilha para realizar a análise, onde podem ser consultados no Apêndice C.

### 5.3.2 **Análise dos Resultados**

A partir dos dados coletados no questionário é Possível verificar o perfil dos participantes. Quanto a instituição de origem de formação, 50% dos entrevistados tiveram sua formação superior em Universidade pública, enquanto que os outros 50% em instituição particular, como mostra a Figura 5-12:

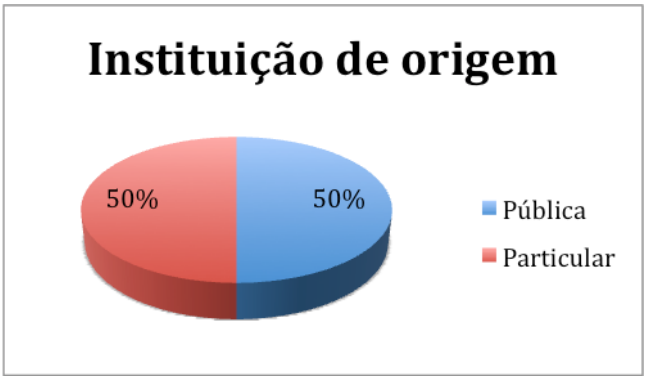


Figura 5-12 – Comparação de custos de defeitos.

Para o tipo de curso, 100% dos entrevistados possuem em sua formação o tipo de curso em Informática/Ciência da computação, como mostra o gráfico da Figura 5-13. Nenhum possui formação em engenharia, matemática ou outro tipo de formação.

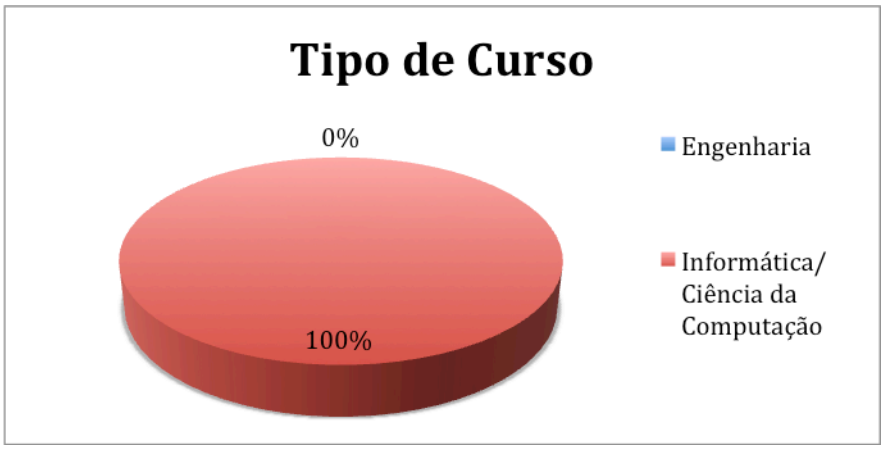


Figura 5-13 – Tipo de curso de formação.

Para o grau de escolaridade, 67% dos entrevistados responderam que possuem seu grau de escolaridade como MBA, enquanto que 33% responderam ser apenas graduados. Não houve ocorrências para respostas de grau de escolaridade em Ensino Médio/Técnico, ou Pós-graduação mostrado na Figura 5-13.

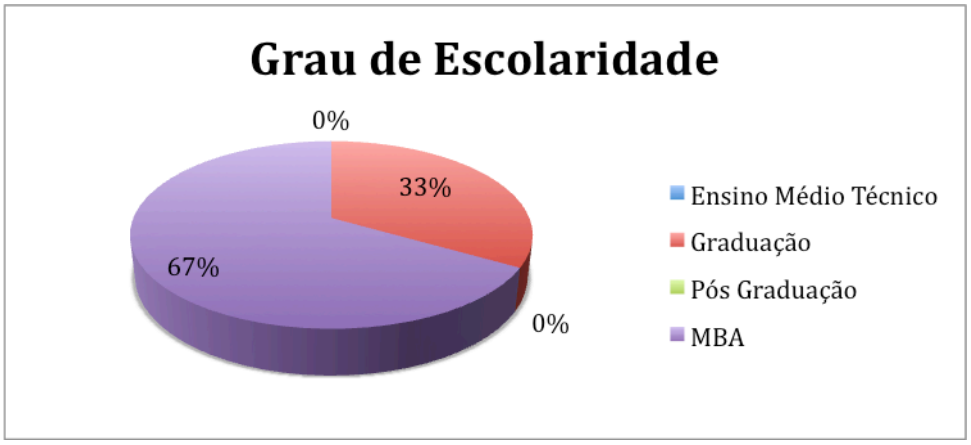


Figura 5-14 – Grau de escolaridade.

A primeira pergunta para definição do perfil de experiência profissional foi quanto a classificação do entendimento sobre teste de software, onde, 17% dos entrevistados classificaram seu entendimento como Excelente, enquanto que 33% classificaram como Alto, 33% como Bom e 17% como Médio. Não houve ocorrências para opções de conhecimento Baixo e Nenhum conhecimento, como mostrados na Figura 5-15.

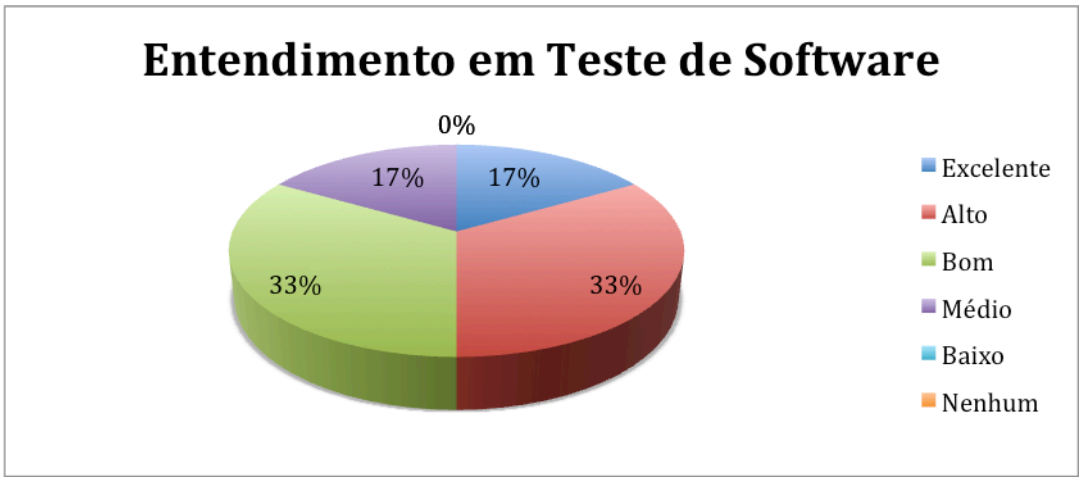


Figura 5-15 – Classificação de entendimento em Teste de Software.

Quando os entrevistados foram questionados sobre experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação todos os entrevistados responderam que foi o cargo de Analista de Teste, não havendo, assim, ocorrência para as opções de Desenvolvedor, Gerente de Projeto, Coordenador de Testes e Analista de Requisitos como mostrada na Figura 5-16.

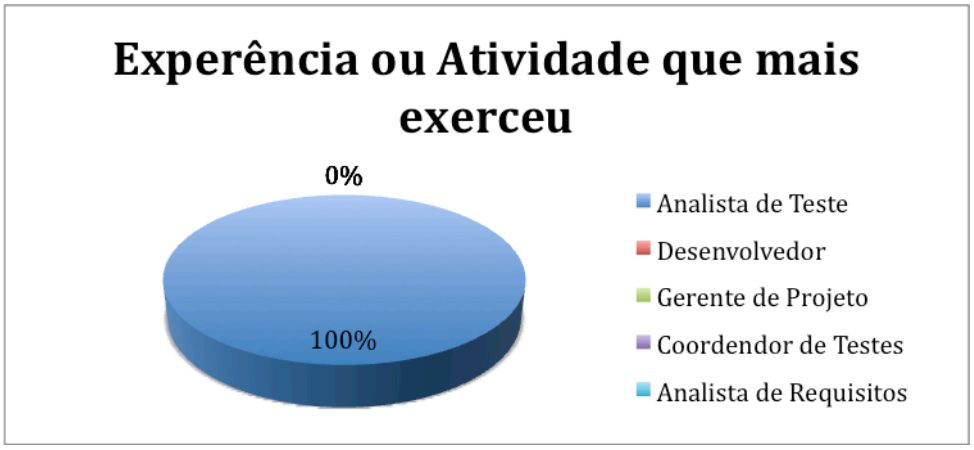


Figura 5-16 – Experiência e Atividade que mais exerceu.

Quanto a experiência em Desenvolvimento de Projetos, quando perguntados sobre quais tipos de sistemas de desenvolvimento já havia participado, 100% dos entrevistados participaram de projetos de Grande Porte, não havendo, assim, ocorrências para as opções de Médio Porte e Pequeno Porte. Como mostrado na Figura 5-17 .


Figura 5-17 – Experiência e Atividade que mais exerceu.

Quando questionados sobre qual área da Engenharia de Software mais haviam atuado todos os entrevistados responderam que foi em Teste de Software, não havendo ocorrências para as demais opções.

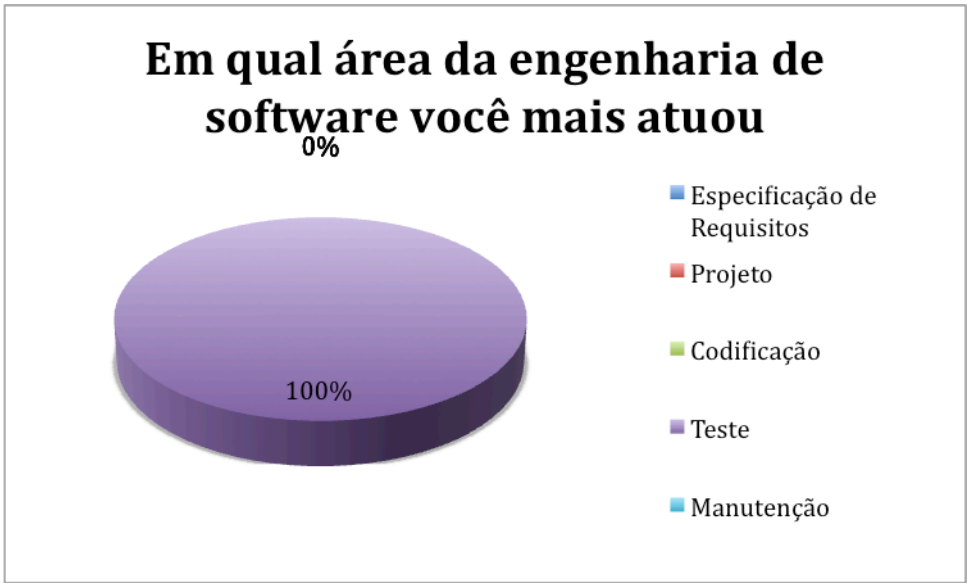


Figura 5-18 – Experiência e Atividade que mais exerceu.

Já quando questionados sobre como se classifica seus entendimentos em homologação, 83% responderam que seu conhecimento em homologação é Bom, enquanto que os outros, 17%, responderam que possuem conhecimento Baixo, não havendo ocorrência para as demais opções, como mostra a Figura 5-19.

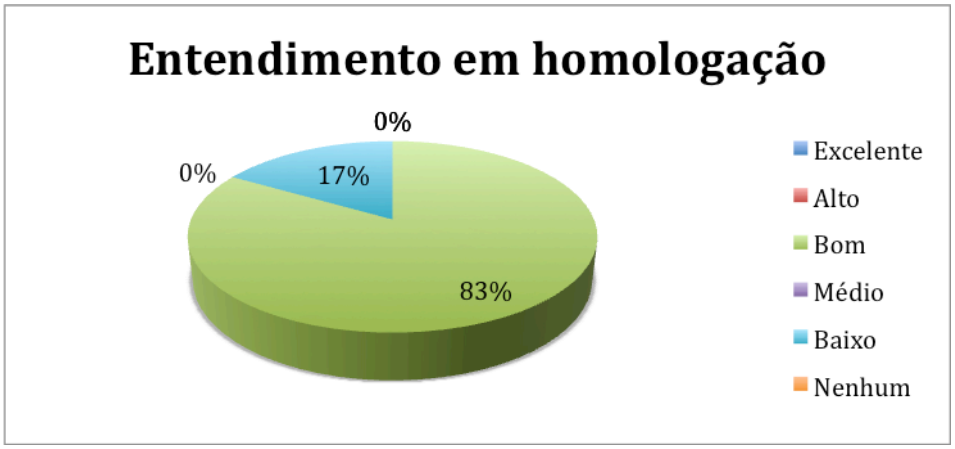


Figura 5-19 – Entendimento em Homologação.

Quando os entrevistados foram questionados de quantos processos de software já haviam participado da definição, todos responderam que já haviam participado de 1 a 2 processos de software, como indica a Figura 5-20.

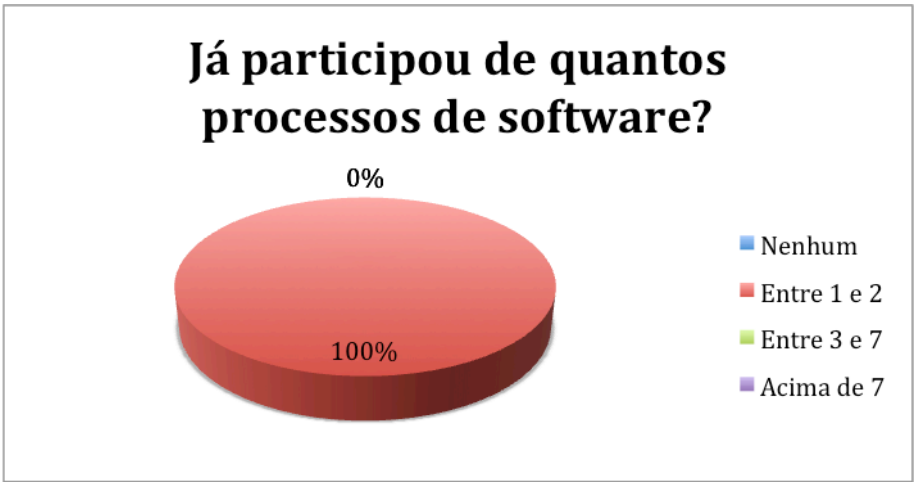


Figura 5-20 – Participação em Processos de softwares.

Aos entrevistados foi questionado também quantos processos já haviam definido, 67% dos entrevistados responderam ter definido entre 1 e 2 processos, enquanto 33% responderam que ainda não haviam definido nenhum processo, mostrado na Figura 5-21.

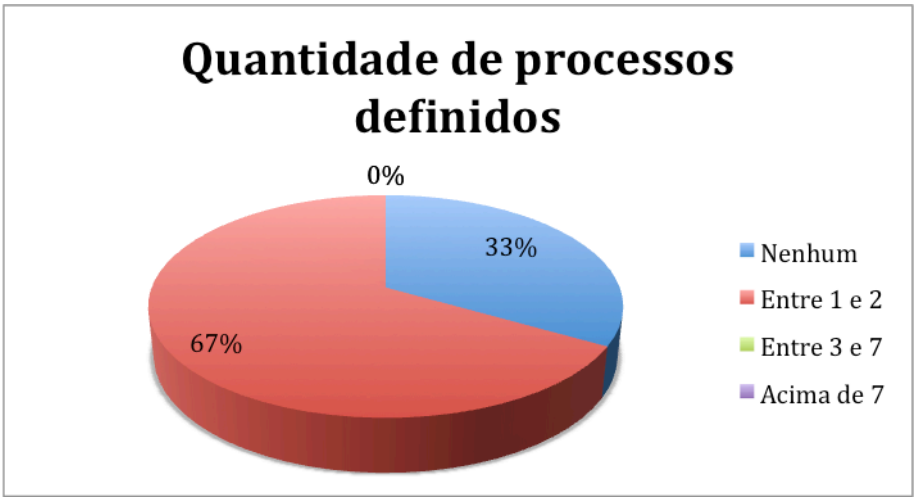


Figura 5-21 – Quantidade de processos definidos.

Quando os participantes foram questionados sobre seu nível de conhecimento em teste de software, 83% responderam que é de nível Bom, enquanto que 17% responderam ser de nível Alto, como mostra a Figura 5-22.

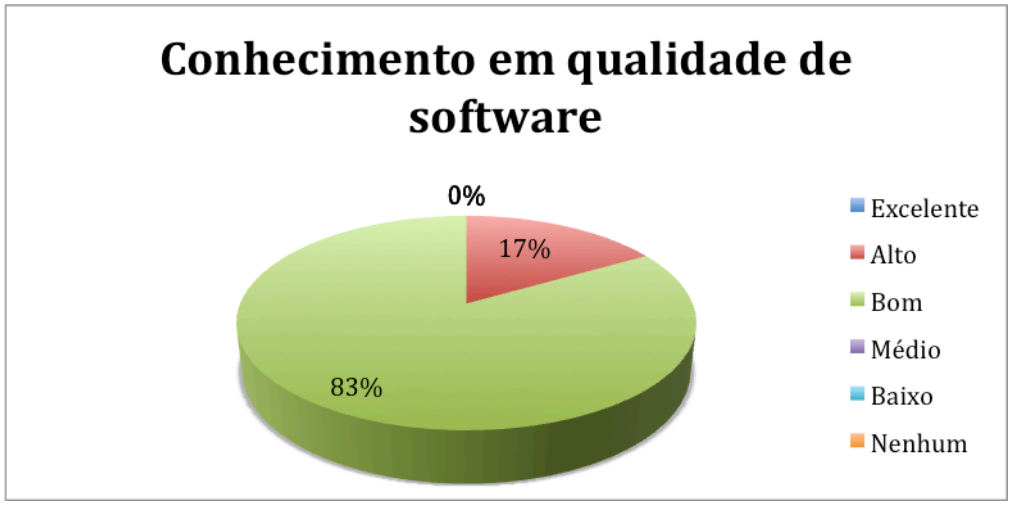


Figura 5-22 – Nível de conhecimento em qualidade de software.

Quando foram questionados quantos modelos ou normas de qualidade já utilizaram, a resposta de todos os entrevistados foi que já haviam utilizado entre 1 e 2 modelos/normas de qualidade. Exibidos na Figura 5-23.

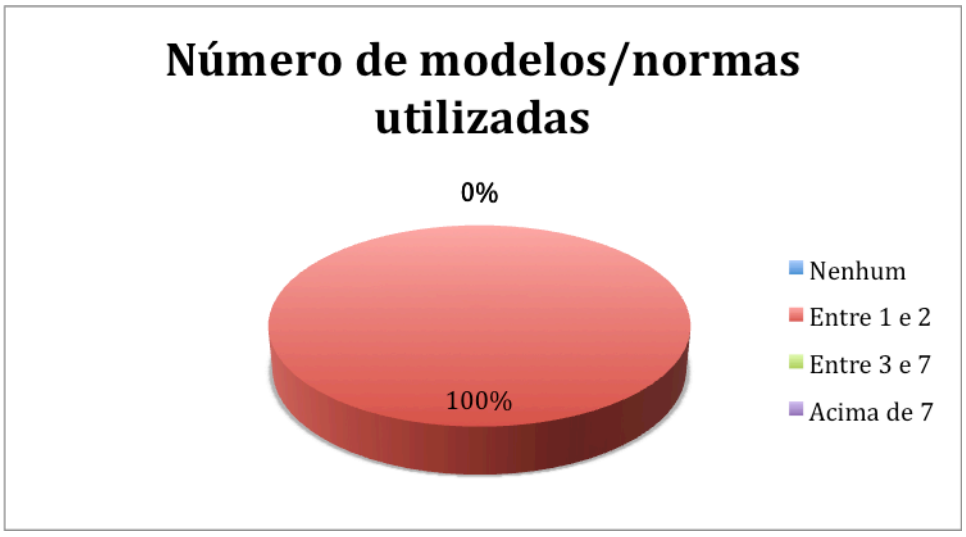


Figura 5-23 – Número de modelos e normas utilizados.

Como resultado da análise do perfil dos participantes do questionário é percebido que a maioria possui uma formação especializada na área de Teste de Software e dizem possuir conhecimento de nível Alto e Bom, desta maneira, essas pessoas representarão um papel critico importante para terceira etapa do questionário

Na segunda etapa do questionário os entrevistados respondem a perguntas relacionadas ao processo e possui como objetivo avaliar a utilização do processo no cenário de aplicação da Fábrica de Software descrita para este experimento, as perguntas serão

relacionadas as atividades do processo e serão três perguntas para cada atividade, Presença, Utilidade e Adequação.

A Tabela 5-1 mostram a consolidação dos resultados das respostas para a característica quanto a presença das atividades no processo.

Tabela 5-1– Itens definidos como presença.

| **Presença/Execução** | |
|---|---|
| **Item** | **Resultado** |
| 1. Não é oferecido pelo processo e não gostaria de tivesse disponível. | **0%** |
| 2. Não oferecido, mas gostaria que tivesse disponível. | **7%** |
| 3. Oferecido, parcialmente. | **9%** |
| 4. Oferecido. | **84%** |
| **Total** | **100%** |

A atividade de Gerar Guia Operacional e Comunicar Cliente é uma atividade desempenhada pelo Coordenador da equipe, onde, em alguns casos há a participação de Analistas de Testes com uma senioridade maior. De maneira que nem toda a equipe é envolvida, apenas seus resultados são apresentados. Por esse motivo, provavelmente, algumas ocorrências para o não conhecimento da atividade, bem como, a ocorrência de que a atividade não é oferecida e necessita de um maior detalhamento.

A atividade de Gerar Plano de Teste, é uma atividade também desempenhada pelo Coordenador de Testes, sendo do conhecimento da equipe apenas seus resultados, o que provavelmente justifica algumas respostas de que a atividade não é oferecida.

As atividades de Gerar *Scripts* de Teste e Executar Scripts de Teste são atividades executadas por Analistas de Testes específicos que tenham habilidades de programação, portando, não são todos que desempenham essas atividades, provavelmente as respostas dos entrevistados que responderam que as atividades não são oferecidas mas gostariam que fossem, tem haver com o pouco contato na realização da atividade.

A Tabela 5-2 mostram a consolidação dos resultados das respostas para a característica quanto a Utilidade das atividades no processo.

Tabela 5-2 – Itens definidos como utilidade.

| **Utilidade** | |
|---|---|
| **Item** | **Resultado** |
| 1. Não é útil. | **0%** |
| 2. Provavelmente é útil, mas ainda não apliquei. | **35.90%** |
| 3. É útil e já apliquei em diferentes ciclos de teste. | **64.10%** |
| **Total** | **100%** |

Segundo os entrevistados 64.10% disseram que as atividades são úteis ao projeto e que já utilizaram em diferentes ciclos de teste. Enquanto que 35.90% dos entrevistados responderam que a atividade provavelmente é útil mas ainda não aplicou.

As atividades referentes a Homologação são executadas apenas por analistas de teste que tenham um perfil específicos não tendo a necessidade de ser aplicada por todos os analistas, provavelmente essas característica das atividades tenham levado a ocorrência de respostas para o item 2. Contribuindo ainda com esta ocorrência as atividades de Gerar Scripts de teste e Executar Scripts de Teste que também são executados por analistas com perfis específicos e as atividades desempenhadas exclusivamente pela coordenação de equipe como Gerar guia Operacional, Gerar Projeto de Teste, Comunicar Cliente e Gerar plano de homologação.

A Tabela 5-3 mostram a consolidação dos resultados das respostas para a característica quanto aos níveis de adequação das atividades no processo.

Tabela 5-3 – Itens definidos como adequação ao nível de detalhamento.

| **Adequação do nível de detalhamento** | |
|---|---|
| **Item** | **Resultado** |
| 1. O detalhamento deve ser aumentado. | **26%** |
| 2. O detalhamento não precisa ser modificado. | **66%** |
| 3. O detalhamento deve ser diminuído. | **8%** |
| **Total** | **100%** |

Um total de 8% dos participantes informaram que 6 atividades precisam ter o seu detalhamento diminuído são elas: Gerar Projeto de Teste, Executar Projeto de Teste, Registrar Defeitos, Homologar Sistema, Registrar Problemas e Analisar Problemas. Com isso infere-se que esses entrevistados conhecem as atividades citadas de maneira mais detalhada. Já 66% acham que o detalhamento das atividades não necessitam ser alterados e 26% acreditam que as atividades deveriam ser mais detalhadas.

## 5.4 Considerações Finais

Este capítulo mostrou como decorreu a execução do processo pós melhorias, apresentou os objetivos a serem atingidos, bem como as métricas que serviram para serem analisadas e se os objetivos estavam sendo atingidos.

O processo após as melhorias atingiram de maneira geral os objetivos definidos pela gerência, SEPG e Coordenação da equipe de teste de software como foi analisado nas métricas qualitativas, bem como satisfizeram as expectativas dos participantes do processo, como mostrou a análise qualitativa. Porém, mesmo com os objetivos apresentados o processo ainda necessita evoluir, aumentando sua maturidade no modelo de processo TMM e aumentando a abrangência dos tipos de testes executados, mas para essa evolução é necessário também que os restantes dos processos evoluam juntos, como por exemplo, o processo de Gerencia de projeto, Gestão de Requisitos, Desenvolvimento de softwares, Gestão de Mudanças e Gestão de Configuração.

# Capítulo 6 **- Conclusão**

O objetivo desta dissertação de mestrado foi apresentar um relato de experiência de melhoria de um processo de Teste de Software em uma fábrica de software. Neste trabalho foi apresentada uma metodologia para melhoria de processo de Teste de Software utilizando a

linguagem de notação BPMN e uma avaliação do processo através de duas formas, a quantitativa e a qualitativa.

Este Capítulo está dividido em 5 (cinco) partes: a seção 6.1 apresenta o sumário do trabalho; na seção 6.2 é apresentada a análise dos resultados obtidos e contribuições; na seção 6.3 são apresentados Possíveis trabalhos futuros; na seção 6.4 são relatadas as limitações deste trabalho.

## 6.1 Sumário do Trabalho

Este trabalho apresentou um estudo de caso de melhoria de processo de teste de software em uma organização de capital misto e a metodologia adotada para alcançar os resultados. Primeiramente, o trabalho apresentou um resumo bibliográfico de Processo de Desenvolvimento de software, apresentado no Capítulo 2 para então apresentar o cenário bibliográfico a respeito de Teste de Software no Capítulo 3.

No Capítulo 6, foi apresentado o ambiente em que o relato de experiência foi realizado e, também, o processo de desenvolvimento de software e o processo de teste de software de versão 1. Após esta apresentação, foram mostrados os pontos fracos, pontos fortes e oportunidades de melhorias identificadas através de análises e reuniões de retrospectiva. Após as análises e, levando em consideração o resultado da análise realizada no processo, o objetivo dos Gestores, SEPG e do coordenador de equipe e, as recomendações do MPS.BR e TMM, foi gerado o Processo de Teste de Software Versão 2.

Após a apresentação do Processo de Teste de software Versão 2, foi apresentada a metodologia utilizada para avaliação, onde através de avaliações qualitativas e quantitativas, baseadas nos interesses das organizações, se chegou a resultados apresentados no Capítulo 5.

Com a evolução do processo de Teste de Software, bem como com a evolução do processo de desenvolvimento de software da fábrica, percebeu-se um aumento da Qualidade do produto entregue ao cliente, aumentando, assim, a sua satisfação. O aumento da satisfação é verificado uma vez diminuindo a ocorrência de defeitos em seu produto, verificando uma diminuição no pedido de mudança e o recebimento de acompanhamento no momento da Homologação desde o início da atividade até o termino.

## 6.2 Contribuições e Resultados Obtidos com o Processo de Teste Versão 2

Os resultados deste trabalho foram armazenados em um repositório de dados disponíveis para os interessados. Após análise, eram divulgados relatórios mensais sobre a produtividade, satisfação dos colaboradores, os pontos fracos, fortes e oportunidades de melhorias do novo processo. Os seguintes resultados foram verificados:

- **Aumento da Produtividade:** com a utilização de ferramentas em determinados momentos no processo, obteve-se um ganho de tempo considerável na execução das atividades. O teste regressivo, por exemplo, contemplado pelas atividades de Gerar/Executar Scripts de Teste, foi umas das atividades que o ganho de produtividade foi mais acentuado;
- **Aumento da Qualidade do Produto:** em função da contratação de uma equipe mais qualificada para área de Teste de Software e o aumento das atividades que compõem o fluxo de funcionamento dos testes, é provável que tenha contribuído para que o sistema tenha se tornando mais estável, atingido o SLA e consideravelmente com menos defeitos nas fases subsequentes ao teste. O aumento da qualidade do produto deu-se também pela institucionalização das áreas de garantia da qualidade e controle da qualidade, com isso, houve uma maior dedicação e expertise voltados para área de Teste, porém, é perceptível que o aumento da qualidade não se resuma somente a essas práticas e sim ao conjunto de melhorias providos ao processo;
- **Sistematização do Processo:** com a sistematização de parte do processo, erros relacionados a desvio de processo ou perda de sincronia entre as equipes foi diminuído consideravelmente para alguns procedimentos e atividades. As atividades de registro de defeitos e problemas, por exemplo, que possui seu *workflow* controlado por uma ferramenta de *Bugtracking* através de uma máquina de estados definida, onde dependendo de categoria o registro é atribuído ao responsável pela próxima tarefa relacionada ao defeito ou problema;
- **Diminuição de descoberta de defeitos em Homologação:** com o aumento da abrangência dos testes em diferentes contextos (tipos de testes), tipos de defeitos que antes não eram detectados no período de teste e sim em homologação passaram a ser encontrados na fase correta (Testes), por exemplo, conformidade com os requisitos, conformidade ao padrão de interface, análise de desempenho, entre outros;
- **Descobertas de defeitos prematuramente:** em função de o Teste de Software começar a acontecer juntamente com o início da codificação do produto, os defeitos acabam

sendo relatados mais “cedo”, ou seja, antes mesmo de começar a execução dos testes de fato e, também, antes do cliente receber o sistema, de maneira a antecipar as correções destes defeitos tanto na especificação quanto na implementação dos requisitos. Isso se dá em função da maioria das inconsistências ser corrigida ainda na iteração de Desenvolvimento do próprio módulo funcional, de modo que não consuma o tempo alocado para o Desenvolvimento de outros módulos tanto para o desenvolvedores, Analistas de Requisitos, Analistas de Testes e Projetistas. Com essas ações o time de desenvolvimento tem a oportunidade de corrigir ou adequar determinadas falhas ou mudanças ainda em tempo de implementação, o que leva também a diminuição de custos e uma quantidade menor de defeitos em fase de homologação e consequentemente produção.

- **Diminuição dos Custos:** a descoberta prematura de defeitos nos módulos funcionais faz com que os mesmos tornem-se menos custosos para a Fábrica de Software tanto no sentido financeiro quanto no sentido moral. No sentido financeiro, pelo fato de os defeitos serem descobertos após a entrega do produto e os envolvidos na correção já estarem alocados em outras tarefas, tendo que parar a realização destas para corrigir os defeitos encontrados, com isso, o tempo de desenvolvimento das novas tarefas era atrasado necessitando de mais esforço alocado para desenvolvê-las, ou seja, gerando mais custos. No sentido moral, pelo fato de um maior número de defeitos serem descobertos antes da entrega do produto para o cliente minimizando os registros de problemas na ferramenta de *Bugtracking*. A análise de redução de custos é baseada na Lei de 10 descrita por Myers;
- **Controle das Atividades Planejadas:** a utilização de ferramentas em determinadas atividades torna o controle de andamento e estimativa de tempo mais exatos, uma vez que a ferramenta dá visibilidade de que ponto, momento ou responsável está determinado procedimento. Este sentimento foi elencado em entrevista com alguns coordenadores das equipes de Teste e Homologação;
- **Melhora da Produtividade dos Analistas de Requisitos:** com os Analistas de Requisitos dedicados apenas a sua função (definida no processo) e não mais dedicando tempo em testes, os mesmos aumentaram sua produtividade na elicitação e especificação de requisitos, dispondo de mais tempo para reuniões com cliente e equipe técnica de desenvolvimento para a especificação mais precisa dos requisitos. A informação de produtividade foi relatada pela gerência de requisitos quanto aos resultados alcançados;
- **Formação de uma Equipe de Testes:** formou-se uma equipe qualificada e dedicada somente para a disciplina de testes e homologação. A equipe atualmente é formada

por um Líder de Testes e 8 Analistas de Testes. O Líder de Testes é responsável basicamente por gerenciar recursos e demandas passadas à equipe, bem como definir juntamente com os analistas os tipos de testes a serem realizados nos módulos funcionais do projeto. Já os Analistas de Testes são responsáveis por gerar o documento Projeto de Teste, a execução dos testes e os registros de defeitos;

- **Redução da quantidade de ciclo de testes:** com alguns ciclos gerados no processo de versão 2, foi percebido que com a adoção das boas práticas discutidas no Capítulo 4, o número de ciclo de testes, para o atendimento ao SLA, foi diminuído. Com essa redução, o esforço da equipe de testes e o tempo que um módulo é liberado para homologação é menor que o antes verificado. Assim, a equipe pode dar vazão em outros módulos ao invés de realizar mais ciclos;
- **Metodologia de definição de custo beneficio:** a metodologia de definição de custo benefício da equipe de teste para a organização, apresentada na métrica M6, pode ser replicada para outras organizações;

Além dos resultados relatados acima, algumas contribuições foram verificados, são elas:

- este trabalho, e seus resultados foram apresentados em forma de artigo e publicado em uma revista científica;
- após a apresentação do processo de teste de versão 2 em um evento local, ocorrido em uma instituição pública, onde, estavam presentes os diretores responsáveis pela área de desenvolvimento de software, um processo semelhante ao Processo de Teste de Software de Versão 2, descrito no Capítulo 4, foi adotado na organização;
- a metodologia adotada para a melhoria do processo pode ser instanciada e aplicada em outra organização, desde que as variáveis inerentes à organização e especificidades do cenários sejam alteradas;

## 6.3 Trabalhos Futuros

Como trabalhos futuros, há necessidade do processo de Teste de Software de Versão 2, sofrer outro ciclo de melhoria a fim de atingir níveis mais elevados de maturidade de processo do TMM e contemplar totalmente as exigências do processo de Verificação e Validação do MPS.BR.

Outro trabalho a ser realizado como evolução do processo de software é implementar de maneira integral, outros tipos de testes que no atual momento são realizados de maneira parcial como: Teste de Banco de Dados, Teste de integração e Teste de Arquitetura. Trazendo assim uma completude maior de testes aos itens alvos e consequentemente o aumento da qualidade do produto.

É importante, também, institucionalizar as métricas que foram utilizadas neste trabalho e criar novas métricas, para que a organização possa constantemente estar medindo variáveis importantes a fim de medir qualidade, custos e andamento do seu processo. Trazendo um monitoramento constante do processo gerando assim subsídios para melhorias contínuas.

## 6.4 Limitações do Processo

Algumas limitações inerentes ao trabalho foram verificadas, são elas:

- **Processo sensível a mudanças:** como o processo de teste é um processo que depende fortemente dos resultados de trabalho de outros processos para seu sucesso, uma alteração nos processos que precedem ao processo de teste, ou um atraso de execução nos mesmos, comprometerão o sucesso do processo de teste de software de versão 2.
- **Processo sensível ao ambiente:** como a metodologia de modelagem e melhoria do processo de teste de software foi voltada para um ambiente particular de uma organização com diversas peculiaridades, como descritas no Capítulo 4, a instanciação deste processo em outros ambientes tem que ser avaliados com foco em características específicas da organização projeto e produto;
- **Tipos de teste:** caso seja colocado um novo tipo de requisito, ou tipo de teste a ser realizado pelo processo, o mesmo terá que passar por evolução, e alteração de procedimentos, atividades e documentos para que os mesmos possam ser contemplados.

# Referências Bibliográficas

BASTOS, A. et al. “Base de conhecimento em teste de software.” 2ª Ed. Niterói. Martins Fontes, 2007.

Berger. P. M. “Instanciação de Processos de Software em Ambientes Configurados na Estação TABA”. Dissertação de Mestrado, COPPE, Universidade Federal do Rio de Janeiro, 2003.

BPMN. “Página oficial da BPMN”, disponível em http://www.bpmn.org/. Acesso em: Janeiro 2011.

CenPRA - Centro de Pesquisas Renato Archer. “Divisão de Melhoria de Processo de Software – DMPS, RT – Processos para Elaboração de Documentos de Teste de Softwares.”. Relatório Técnico, 2001.

Chrissis, M.B., M. Konrad, and S. Shrum. “CMMI: Guidelines for Process Integration and. Product Improvement.” ,Second Edition ed., 2006.

CMMI. “Software Engineering Institute - CMM (CAPABILITY MATURITY MODEL)”, disponível em http://www.sei.cmu.edu. Acesso em 2011.

Cobra. “Página oficial da Cobra Tecologia”, disponível em http://www.cobra.com.br/. Acesso em: Janeiro 2011.

Ericson, T.; Subotic, A; Ursing S. “Towards a Test Improvment Model”. Proceedings of the Fourth European Conference on Software Testing, Analysis & Review Amsterdam, 2009.

Falbo, R. A.; Menezes, C. S. de; Rocha, A. R. C. “Assist-Pro: um assistente inteligente para apoiar a definição de processos de software”, Anais do XII Simpósio Brasileiro de Engenharia de Software. Florianópolis, SC, 1999.

FIlho, T. R. M.; Rios, E. “Projeto e Engenharia de Software: Teste de Software”. 1ª, ed. Rio de Janeiro. Alta Books. 2002.

Hetzel, William. “Guia Completo ao Teste de Software. Editora Campus, Rio de Janeiro, 1987.

Humphrey, W. S. “Managing the Software Process, SEI Series in Software Engineering.”. Addison-Wesley Publishing Company. 1989.

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

ISO 9000. “Quality Management and Quality Assurance Standards – part 3: guidelines for the application of ISO 9001:1994 to the development, supply and maintenance of computer software”. International Organization fot Standartization, 1991.

ISO/IEC 15504. “:  Information technology – software process assessment. International Organization for Standardization”. International Organization for Standartization, Geneve. 2008.

ISO/IEC-12207. “Systems and Software engineering – Software life cycle processes”. International Organization for Standartization (ISO). Geneve. 2008.

ISO/IEC 9126-1, “Engenharia de Software – Qualidade de Produto - Modelo de Qualidade”, Associação Brasileira de Normas Técnicas, 2003.

Koscianski, A.; Soares, M. S. “Qualidade de Software”. 2ª ed. São Paulo.  Novatec Editora, 2007.

Lima, A.M.; et al. “WebAPSEE: Um Ambiente Livre e Flexível Para Gerência de Processos de Software.” In: VII Workshop sobre Software Livre, 2006, Porto Alegre. Fórum Internacional de Software Livre 7.0, 2006. p. 163-168.

Machado, L.F.D.C., Santos, G., Oliveira, K.M., Rocha, A.R.C., Def-Pro: “Uma ferramenta para apoiar a definição do processo padrão”. Anais da XI Conferência Internacional de Tecnologia de Software – XI CITS, Curitiba, 2000.

Maidantchik, C. L. L. M. “Gerência de Processos de Software para Equipes Geograficamente Dispersas.” COPPE/UFRJ, no. Tese de D.Sc. Tio de Janeiro. 1999.

Mcfeeley, B. “A User´s Guide for Software Process Improvment”. Software Enginnering Institute Handbook. Carnegie Mellon University. CMU/SEI-96-HB-001, 1996.

Molinari, L. “Teste de Software”.1ª Edição. ISBN. 857194959X. Florianópolis. Editora Erica. 2006.

MYERS, Glenford J., “The Art of Software Testing”, 1979.

Oliveira, S. R. B. “Processo de Software: Princípios, Ambientes e Mecanismos de execução”. Exame de Qualificação do Programa de Doutorado do CIN/UFPE. Recife. 2006.

Oliveira, S. R. B.; Vasconcelos, A. M. L.; “Publicação do Processo de Software no ImPProS: Um Ambiente de Implementação Progressiva do Processo de Software”. Revista Traços vol-10, n. 16. Belém. 2006.

Paulk, M. C.; Curtis, B.; Chrissis. M. B.; Weber. C. V.; “Capability Maturity Model for Software, Version 1.1”. Techinal Report CMU/SEI-93-TR-024. Software Engineering Institute – Carnegie Mellon University. 1993.

Pfleeger, S. L. “Software Engineering: theory and pratice”. 2nd edition. Rentice Hall. Inc., ISBN 0-13-029049-1. 2004.

Pressman, R. S. “Software engineering – A practitional approach.”. 5th ed. McGraw Hill 2007.

Reis, R. Q.; Reis, C. A. L. “Engenharia de Software, Notas de Aula da Especialização em Análise de Sistemas.”. DI-UFPA. 1999.

Rios, E.; Trayahu R. M. F. “Teste de Software”. 2ª Edição. ISBN 8576081288. Alta Books. 2006.

Rocha. A. R. C.; Maldonado, J. C.; Weber, K. C. “Qualidade de Software: Teoria e Prática”. São Paulo. Pretince Hall. 2001.

Ryan K. L. Ko, Stephen S. G. Lee, Eng Wah Lee. “Business Process Management (BPM) Standards: A Survey”. In: Business Process Management Journal, Emerald Group Publishing Limited. Volume 15 Issue 5. ISSN 1463-7154. 2009

Silva, E. L.; Menezes, E. M. “Metodologia da Pesquisa e Elaboração da Dissertação”. 3. ed. rev. Atual – Laboratório de Ensino a Distância da UFSC. Florianópolis, Brasil. 2001.

Softex - Sociedade para Promoção da Excelência do Software Brasileiro. “MPS.BR - Melhoria de Processo do Software Brasileiro”. Guia Geral, versão 1.1, 2011.

SOGETI. “TPI – Test Process Improvment.”. Disponível em http://www.ie.spgeti.com/. Acesso em Janeiro. 2011.

Somerville. “Engenharia de Software”. São Paulo: Prentice-Hall, 2007.

Weber, K. C. et AL.: “Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira.”. Conferência Latinoamericana de Informática – CLEI, Arequipa-Peru, 2004.

Welzel, D., HAUSEN, H.; SCHMIDT, W., “Tailoring and Conformance Testing of Software Processes: The ProcePT”. Approach, In Proc. IEEE Software Engineering Standards Symposium, 1995.

Zahran, S. “Software Process Improvement”, Addison Wesley Longman Inc, 1998.

# Apêndice A

# Métricas Definidas

Para que seja necessário avaliar de forma quantitativa o Processo de Teste, fez-se necessária a utilização de métricas de acordo com o GQM.

## A.1. Definição detalhada das métricas

De acordo com o Objetivo 1 "Definir a produtividade da equipe de desenvolvimento utilizando as práticas definidas no processo" foi definida uma métrica como parâmetro para saber se o objetivo foi atendido. Abaixo o Quadro 6-1– Detalhamento da métrica M1 . que detalha a métrica M1:

Quadro 6-1– Detalhamento da métrica M1 .

| **Métrica** | **M1:** Porcentagem de defeitos encontrados na fase de Teste x em fase de Homologação |
|---|---|
| **Descrição** | São levantados todos os defeitos encontrados para uma determinada parte do sistema (módulo funcional), onde é retirado a porcentagem de defeitos que foram relatados pela equipe de testes, antes do sistema ser entregue em homologação. |
| **Mnemônimo** | PDPIxPDPM |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • RT = ((DEFT x 100) / TDT);<br>• RH = ((DETf x 100) / TDH);<br>• Se RT < RH, então V Se não F; |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Porcentagem. |
| **Composição da métrica** | • DEFT = Defeitos Encontrados na Fase de Teste;<br>• DEFH = Defeitos Encontrados na Fase de |

| **Métrica** | **M1:** Porcentagem de defeitos encontrados na fase de Teste x em fase de Homologação |
|---|---|
| | Homologação;<br>• TDT = Total de Defeitos em fase de Teste;<br>• TDH = Total de Defeitos em Homologação;<br>• RT = Resultado de Teste<br>• RH = Resultado de Homologação; |
| **Procedimento análise** | Os registros serão obtidos e contabilizados na ferramenta de *bugtraking* utilizada pela organização. A porcentagem será comparada no total dos defeitos (trivial, importante e crítico), onde será gerado um gráfico de pizza comparativo entre as criticidades/quantidade de defeitos encontrados na fase de teste e fase de homologação. |
| **Forma de apresentação** | Gráfico de Pizza. |

De acordo com o Objetivo 2 "Quantificar completude na cobertura de Testes" foram definidas duas métricas como parâmetro para saber se o objetivo foi atendido. Abaixo o Quadro 6-2 que detalha a métrica M2 e 6-3 que detalha a métrica M3:

Quadro 6-2 – Resultado da M2: Quantificar a velocidade da equipe de desenvolvimento.

| **Métrica** | **M2:** Quantificar Cobertura real da Equipe de Testes. |
|---|---|
| **Descrição** | Quantificar a cobertura real da equipe de teste quanto aos ativos do constantes do processo de testes que deveriam ser testados. |
| **Mnemônimo** | QICP |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • QIT = ∑IT;<br>• QIL = ∑IL;<br>• QIP = ∑IP;<br>• QIN =∑IN. |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • QIT =Quantidade de Itens Totalmente atendidos;<br>• QIL = Quantidade de Itens Largamente atendidos;<br>• QIP = Quantidade de Itens Parcialmente Atendidos;<br>• QIN = Quantidade de Itens Não atendidos;<br>• ∑IT = Somatória de Itens Totalmente atendidos;<br>• ∑IL = Somatória de Itens Largamente atendidos;<br>• ∑IP = Somatória de Itens Parcialmente atendidos;<br>• ∑IN = Somatória de Itens Não atendidos. |
| **Procedimento análise** | A avaliação é realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação, onde, através de entrevistas e auditorias em |

| **Métrica** | **M2:** Quantificar Cobertura real da Equipe de Testes. |
|---|---|
| | ativos do processo foi avaliado se a execução dos testes estavam de fato obtendo uma completude nos itens alvos de testes. |
| **Forma de apresentação.** | Gráfico de Barras. |

A segunda métrica tem o objetivo 2 de quantificar atividades e boas praticas que não estão sendo executadas, como mostrado no Quadro 6-3.

Quadro 6-3 – Detalhamento da métrica M3.

| **Métrica** | **M3:** Quantificar Grau de atendimento as atividades descritas. |
|---|---|
| **Descrição** | Quantificar o grau de atendimento as Atividades e boas práticas listadas no processo. As atividades podem ser classificadas em T, L, P ou N. |
| **Mnemônimo** | QICP |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • QAT = ∑AT;<br>• QAL = ∑AL;<br>• QAP = ∑AP;<br>• QAN =∑AN. |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • QAT = Quantidade de Atividades Totalmente executadas;<br>• QAL = Quantidade de Atividades Largamente executadas;<br>• QAP = Quantidade de Atividades Parcialmente executadas;<br>• QAN = Quantidade de Atividades Não executadas;<br>• ∑AT = Somatória de Atividades Totalmente executadas;<br>• ∑AL = Somatória de Atividades Largamente executadas;<br>• ∑AP = Somatória de Atividades Parcialmente executadas;<br>• ∑AN = Somatória de Atividades Não executadas. |
| **Procedimento análise** | A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação. Onde através de entrevistas e auditorias em ativos do processo realizadas foram coletadas as métricas. |
| **Forma de apresentação.** | Gráfico de Barras. |

De acordo com o Objetivo 3 “Abrangência ao Modelo de Referencia MPS.BR” foi definida uma métrica como parâmetro para saber se o objetivo foi atendido. Abaixo no Quadro 6-4 que detalha a métrica M4.

Quadro 6-4 – Detalhamento da métrica M4.

| **Métrica** | **M4:** Quantificar Grau de atendimento as atividades descritas. |
|---|---|
| **Descrição** | Quantificar o grau de atendimento as Atividades e boas práticas listadas no processo as exigências de VER e VAL do MPS.BR. As atividades podem ser classificadas em T, L, P ou N. |
| **Mnemônimo** | QICP |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • QAT = ∑AT;<br>• QAL = ∑AL;<br>• QAP = ∑AP;<br>• QAN =∑AN. |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • QAT = Quantidade de Atividades Totalmente executadas;<br>• QAL = Quantidade de Atividades Largamente executadas;<br>• QAP = Quantidade de Atividades Parcialmente executadas;<br>• QAN = Quantidade de Atividades Não executadas;<br>• ∑AT = Somatória de Atividades Totalmente executadas;<br>• ∑AL = Somatória de Atividades Largamente executadas;<br>• ∑AP = Somatória de Atividades Parcialmente executadas;<br>• ∑AN = Somatória de Atividades Não executadas. |
| **Procedimento análise** | A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação. Onde através de entrevistas e auditorias em ativos do processo realizadas foram coletadas as métricas. |
| **Forma de apresentação.** | Gráfico de Barras. |

De acordo com o Objetivo 4 “Definir se os tipos de testes projetados para serem executados cobrem as regras de negócio e os requisitos contemplados no projeto” foi definida uma métrica como parâmetro para saber se o objetivo foi atendido. Abaixo no Quadro 6-5 que detalha a métrica M5.

Quadro 6-5 – Detalhamento da métrica M5.

| **Métrica** | **M5:** Quantificar o número de atividades executadas. |
|---|---|
| **Descrição** | Quantificar o grau de atendimento as Atividades e boas práticas listadas no processo. As atividades podem ser classificadas em T, L, P ou N. |
| **Mnemônimo** | QGAMPS |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • QAT = ∑AT;<br>• QAL = ∑AL;<br>• QAP = ∑AP;<br>• QAN =∑AN. |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • QAT = Quantidade de Atividades Totalmente executadas;<br>• QAL = Quantidade de Atividades Largamente executadas;<br>• QAP = Quantidade de Atividades Parcialmente executadas;<br>• QAN = Quantidade de Atividades Não executadas;<br>• ∑AT = Somatória de Atividades Totalmente executadas;<br>• ∑AL = Somatória de Atividades Largamente executadas;<br>• ∑AP = Somatória de Atividades Parcialmente executadas;<br>• ∑AN = Somatória de Atividades Não executadas. |
| **Procedimento análise** | A avaliação foi realizada ao final da execução do processo de teste, juntamente com a passagem do sistema para homologação. Onde através de entrevistas e auditorias em ativos do processo realizadas foram coletadas as métricas. |
| **Forma de apresentação.** | Gráfico de Barras. |

De acordo com o Objetivo 5 “possui como objetivo revelar a gerência se é vantajoso continuar investindo e uma equipe de teste para a organização” foi definida uma métrica como parâmetro para saber se o objetivo foi atendido. Abaixo Quadro 6-6 detalha a métrica M6.

Quadro 6-6 – Detalhamento da métrica M6.

| **Métrica** | **M6:** Quantificar Grau de atendimento as atividades descritas. |
|---|---|
| **Descrição** | Aplicando a lei 10 de Myers quantificar se vale a pena a organização continuar investindo em uma equipe de teste ou tratar os erros apenas quando forem reportados? |
| **Mnemônimo** | CBET |
| **Atomicidade** | Não atômica. |

| **Métrica** | **M6:** Quantificar Grau de atendimento as atividades descritas. |
|---|---|
| **Equação de Cálculo** | • CCDT = NDT * TMC * CC;<br>• CCDM = NDM * TMC * CC;<br>• CCDC = NDC * TMC * CC;<br>• CTCE = CCDT + CCDC + CCDM;<br>• CTCT = TMTT * CT * NUC;<br>• CTCM = TMTM * CT * NUC;<br>• CTC = TMTC * CT * NUC;<br>• CTT = CTCT + CTCM + CTC;<br>• CETCD = CTCE * CTT;<br>• CTCFH = CTC * LM<br>• CTCFP = CTCFH * LM |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • CTCE = Custo Total de Correção Estimado;<br>• CETCD = Custo Estimado de Testes com Correção de Defeitos;<br>• CTCFP = Custo Total de Correção em Fase de Produção;<br>• LM = Lei de Myers;<br>• CTCFH = Custo Total de Correção em Fase de Homologação;<br>• CCDT = Custo de Correção dos Defeitos Triviais;<br>• CCDM = Custo de Correção de Defeitos Medios<br>• CCDC = Custo de Correção de Defeitos Criticos;<br>• NDT = Numero de Defeitos Triviais;<br>• NDM = Numero de Defeitos Médios;<br>• NDC = Numero de Defeitos Críticos;<br>• TMC = Tempo Médio de Correção;<br>• CC = Custo de Correção por Hora Trabalhada.<br>• CTT = Custo Total de Teste;<br>• TMTM = Tempo Médio de Teste de use case Médios;<br>• TMTT = Tempo Médio de Teste de use cases Triviais;<br>• CT = Custo de Teste;<br>• NUC = Numero de U.C testados;<br>• CTCT = Custo de Teste de Complexidade Trivial;<br>• CTCM = Custo de Teste de Complexidade Media;<br>• CTC = Custo de Teste Complexo. |
| **Procedimento análise** | O MPS.BR adota um padrão de pontuação onde são dadas três pontuações: T, L, P, N e NA. Onde a pontuação T significa que o processo atente Totalmente ao item exigido, L significa que o item é atendido Largamente, P significa que o item é atendido Parcialmente o N significa que o item não é atendido e o NA significa que o item Não se aplica ao processo.<br>O resultado da avaliação para o processo de Validação dos sete itens exigidos pelo MPS.BR todos foram receberam T, ou seja, Totalmente aderente ao MPS.BR. |
| **Forma de apresentação.** | Gráfico de Barras. |

De acordo com o Objetivo 6 “avaliar a aderência dos ativos constantes no processo de teste ao modelo de maturidade TMM” foi definida uma métrica como parâmetro para saber se o objetivo foi atendido. Abaixo o Quadro 6-7 detalha a métrica M7.

Quadro 6-7 – Detalhamento da Métrica M7.

| **Métrica** | **M7:** Medir grau de aderência da maturidade do processo em relação as boas práticas definidas no modelo TMM |
|---|---|
| **Descrição** | Verificar em que nível se encontra o processo de teste de software quando comparado ao TMM. |
| **Mnemônimo** | MGMPT |
| **Atomicidade** | Não atômica. |
| **Equação de Cálculo** | • MGMPT = QATLA; |
| **Valor base** | 1.00 |
| **Unidade de Medida** | Unidade. |
| **Composição da métrica** | • QATLA = Quantidade de Objetivos Totalmente ou Largamente Atendidos. |
| **Procedimento análise** | Ao final da execução do processo de teste de software, foi realizado um levantamento e avaliação das atividades e ativos do processo afim de realizar a aderência ao processo de melhoria TMM e com isso revelar o nível de maturidade do processo de teste de software da organização. A avaliação foi realizada por um avaliador autorizado da SOFTEX. O método de avaliação foi o mesmo utilizado para verificar a aderência ao MPS.BR descrito na métrica M4. |
| **Forma de apresentação.** | Gráfico de Barras. |

# Apêndice B

# Resultado da coleta das métricas

O Quadro 6-8 apresenta o período em que as métricas foram coletadas.

Quadro 6-8 – Período em que as métricas foram coletadas.

| Períodos | | | |
|---|---|---|---|
| | **Início** | **Fim** | **Quantidade de dias** |
| Entrega 1 | 06/08/2010 | 16/09/2010 | 40 |
| Entrega 2 | 17/09/2010 | 27/10/2010 | 40 |
| **Total** | | | **80** |

A Quadro 6-9 apresenta os Defeitos encontrados na Entrega 1 (E1) e na Entrega 2 (E2), orientados por fase (Homologação e Testes) e Criticidade.

Quadro 6-9 – Resultado da M1: Número de Defeitos descobertos na fase de Teste e de Homologação para E1 e E2.

| | **Teste** | | | **Homologação** | | |
|---|---|---|---|---|---|---|
| | Trivial | Médio | Critico | Trivial | Médio | Critico |
| **Entrega 1** | 46 | 170 | 22 | 0 | 0 | 4 |
| **Entrega 2** | 101 | 316 | 47 | 7 | 7 | 5 |

O Quadro 6-10 apresenta os Itens Alvo de testes descritos no Plano de Teste, seu grau de atendimento e a justificativa para sua classificação utilizados na M2.

Quadro 6-10 – Resultado da M2: Comparação dos resultados dos projetos de testes descritos com os alvos de testes.

| Nº | Itens Alvo | Atendimento | Observação |
|---|---|---|---|
| 1 | Código Fonte | Largamente | Pelo fato do Teste ser uma estratégia caixa preta, o mesmo testa o código fonte através de funcionalidades e não através de testes de caixa branca. |
| 2 | Especificação de Requisito | Totalmente | Os projetos de Testes elaborados com base nas especificações de requisitos e regras de negócio. |
| 3 | Dicionário de dados | Largamente | Os projetos de testes testam diretamente apenas os registros de entradas e saídas, não sendo alvo central os campos de uso interno do sistema. |
| 4 | Integração Externa | Parcialente | As integrações com sistemas externos são testadas apenas seus domínios e sua disponibilidade no Sistema central. A integração propriamente dita não é alvo de teste. |
| 5 | Regra de Negócio | Totalmente | As regras de negócio são testadas . |
| 6 | Padrão de Interface | Parcialente | O teste de interface é ralizado de maneira superficial, fcando em comportamento de campos, não sendo foco principal a aderência total de cores e formas. |
| 7 | Arquitetura do sistema | Parcialente | |
| 8 | Banco de Dados | Largamente | |

O Quadro 6-11 consolida a contagem dos Itens alvo de Testes e orienta por grau de atendimento para M3.

Quadro 6-11 – Resultado da M2: Resumo de atendimento.

| Totalmente | Largamente | Parcialmente |
|---|---|---|
| 2 | 3 | 3 |

O Quadro 6-12 consolida os resultados dos projetos de testes descritos com as atividades do processo e orienta por grau de atendimento.

Quadro 6-12 – Resultado da M4 Comparação dos resultados dos projetos de testes descritos com as Atividades do processo.

| Nº | Atividades Contidas no processo | Atendimento |
|---|---|---|
| | | |
| 1 | Gerar/Rever Guia Operacional de Testes | Largamente |
| 2 | Atividade Gerar/Rever Planos: Gerar/Rever Plano de Teste | Largamente |

| Nº | Atividades Contidas no processo | Atendimento |
|---|---|---|
| 3 | Atividade Gerar Planos: Gerar/Rever Plano de Homologação. | Largamente |
| 4 | Gerar/Rever Projeto de Teste | Totalmente |
| 5 | Executar Projeto de Teste/Scripts de Teste | Totalmente |
| 6 | Registrar Defeitos | Totalmente |
| 7 | Gerar/Rever *Scripts* de Teste | Totalmente |
| 8 | Liberar *Build* | Parcialmente |

A Quadro 6-13 consolida a contagem das atividades e orienta por grau de atendimento para M4.

Quadro 6-13 – Resultado da M3: Avaliar a complexidade das atividades.

| Totalmente | Largamente | Parcialmente |
|---|---|---|
| 4 | 3 | 1 |

Os Quadros 6-14 e 6-15 mostram os resultados da avaliação para os processos de VAL e VER.

Quadro 6-14 – Resultado da M4: Avaliação da Aderência quanto a área de processo de VAL do MPS.BR

| **Validação** | | |
|---|---|---|
| **PREENCHIDO PELA EMPRESA** | | |
| **Resultado esperado / evidências** | **Fonte da evidência** | **Final** |
| **O propósito do processo Validação é confirmar que um produto ou componente do produto atenderá a seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido.** | | |
| **VAL 1. Produtos de trabalho a serem validados são identificados.** As evidências apresentadas para este resultado permitem assegurar que foram identificados os produtos de trabalho a serem validados? | | |
| No documento de plano de Homologação a seção 4 - Itens Alvo de Homologação | *Plano de Homologação* | *T* |
| | (T,L,P,N,NA) | T |
| **VAL 2. Uma estratégia de validação é desenvolvida e implementada, estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação.** As evidências apresentadas para este resultado permitem assegurar que foi desenvolvida e implementada uma estratégia de validação que define o cronograma, participantes envolvidos, métodos para validação e o material a ser utilizado na validação? | | |
| Estratégia Desenvolvida - Documento de Plano de Homologação seção 5. | Plano de Homologação | T |

| Validação | | |
|---|---|---|
| **PREENCHIDO PELA EMPRESA** | | |
| **Resultado esperado / evidências** | **Fonte da evidência** | **Final** |
| Participantes envolvidos - Planilha de Acompanhamento | Plano de Homologação | T |
| Material a ser utilizado na validação - Email identificando a baseline a ser utilizada no ciclo de Homologação. | Email | T |
| Entrega Formal do Sistema ao Cliente via Mídia eletrônica contendo todos os insumos para homologação. | Carta Formal | T |
| | (T,L,P,N,NA) | T |
| **VAL 3. Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido.**<br>As evidências apresentadas para este resultado permitem assegurar que foram identificados critérios e procedimentos a serem utilizados para a validação dos produtos de trabalho, bem como permitem assegurar que foi estabelecido um ambiente para validação? | | |
| Critérios e Procedimentos - Projeto de Teste | Projeto de Teste | T |
| Ambiente de Homologação - Banco de dados dedicados, instância de servidor de aplicação dedicada. | Servidor de aplicação e Banco de dados | T |
| | (T,L,P,N,NA) | T |
| **VAL 4. Atividades de validação são executadas para garantir que o produto esteja pronto para uso no ambiente operacional pretendido.**<br>As evidências apresentadas para este resultado permitem assegurar que as atividades de validação foram executadas para garantir que o produto esteja pronto para uso no ambiente operacional pretendido? | | |
| Carta de aceitação emitida pelo cliente | Carta do cliente | T |
| O registros obtidos da ferramenta de Bug Tracking | Mantis | |
| | (T,L,P,N,NA) | T |
| **VAL 5. Problemas são identificados e registrados.**<br>As evidências apresentadas para este resultado permitem assegurar que os problemas identificados durante as atividades de validação foram registrados? | | |
| Ferramentas de Bug tracking | Mantis | T |
| | (T,L,P,N,NA) | T |
| **VAL 6. Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas.**<br>As evidências apresentadas para este resultado permitem assegurar que os resultados das atividades de validação foram analisados e disponibilizados para as partes interessadas? | | |
| Histórico do atendimento e alteração do status dos registros na Ferramentas de Bug tracking | Mantis | T |
| | (T,L,P,N,NA) | T |
| **VAL 7. Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas.**<br>As evidências apresentadas para este resultado permitem assegurar que os produtos de software desenvolvidos estavam prontos para o uso pretendido? | | |

| Validação | | |
|---|---|---|
| **PREENCHIDO PELA EMPRESA** | | |
| **Resultado esperado / evidências** | **Fonte da evidência** | **Final** |
| Carta de Aceitação emitida pelo Cliente | Carta oficial | T |
| Atendimento ao SLA de Homologação | Email | |
| | (T,L,P,N,NA) | T |

Quadro 6-15 – Resultado da M4: Avaliação da Aderência quanto a área de processo de VER do MPS.BR

| Verificação | | |
|---|---|---|
| **PREENCHIDO PELA EMPRESA** | | |
| **Resultado esperado / evidências** | **Fonte da evidência** | **Final** |
| **O propósito do processo Verificação é confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto atende apropriadamente os requisitos especificados.** | | |
| **VER 1. Produtos de trabalho a serem verificados são identificados.**<br>As evidências apresentadas para este resultado permitem assegurar que os produtos de trabalho sujeitos à verificação foram identificados? | | |
| No documento de plano de teste, a seção 4 - Itens Alvo de Teste | *Plano de Teste* | *T* |
| | (T,L,P,N,NA) | T |
| **VER 2. Uma estratégia de verificação é desenvolvida e implementada, estabelecendo cronograma, revisores envolvidos, métodos para verificação e qualquer material a ser utilizado na verificação.**<br>As evidências apresentadas para este resultado permitem assegurar que há uma estratégia definida para a verificação que inclua cronograma, revisores envolvidos, métodos a serem utilizados e materiais a serem empregados na verificação? | | |
| Estratégia Desenvolvida - Documento de Plan de Teste, seção 5. | Plano de Teste | T |
| Participantes envolvidos - Planilha de Acompanhamento | Planilha de acompanhamento | T |
| Material a ser utilizado na validação - Email identificando a TAG a ser utilizada no ciclo e desenvolvimento. | Email | T |
| | (T,L,P,N,NA) | T |
| **VER 3. Critérios e procedimentos para verificação dos produtos de trabalho a serem verificados são identificados e um ambiente para verificação é estabelecido.**<br>As evidências apresentadas para este resultado permitem assegurar que: (i) critérios e procedimentos a serem utilizados para a verificação foram identificados?; (ii) foi estabelecido um ambiente para verificação? | | |
| Critérios e Procedimentos - Projeto de Teste | Projeto de Teste | T |

| Verificação | | |
|---|---|---|
| **PREENCHIDO PELA EMPRESA** | | |
| **Resultado esperado / evidências** | **Fonte da evidência** | **Final** |
| Ambiente de Teste - Banco de dados dedicados, instância de servidor de aplicação dedicada. | Servidor de aplicação e Banco de dados | T |
| | (T,L,P,N,NA) | T |
| **VER 4. Atividades de verificação, incluindo testes e revisões por pares, são executadas.**<br>As evidências apresentadas para este resultado permitem assegurar que as atividades de verificação, incluindo tanto testes quanto revisão por pares, foram executadas de acordo com o planejado? | | |
| Relatório de Teste - Evidência de Execução | Relatório de Teste | P |
| O registros obtidos da ferramenta de Bug Tracking | Mantis | |
| | (T,L,P,N,NA) | P |
| **VER 5. Defeitos são identificados e registrados.**<br>As evidências apresentadas para este resultado permitem assegurar que os defeitos identificados durante as atividades de verificação foram identificados e registrados? | | |
| Relatório de Testes | Relatório de Teste | T |
| Registros das Ferramentas de Bug tracking | Mantis | T |
| | (T,L,P,N,NA) | T |
| **VER 6. Resultados de atividades de verificação são analisados e disponibilizados para as partes interessadas.**<br>As evidências apresentadas para este resultado permitem assegurar que os resultados das atividades de verificação foram (i) analisados? (ii) foram adotados procedimentos para disponibilização dos resultados para as partes interessadas? | | |
| Documentos de Relatório de Testes publicados em Ferramenta de Controle de Versão interna. | CVS | T |
| Email enviado aos gerentes contendo o resumo de defeitos encontrados por U.C. | Email | L |
| Histórico do atendimento e alteração do status dos registros na Ferramentas de Bug tracking | Mantis | |
| | (T,L,P,N,NA) | L |

Os Quadros 6-16, 6-17, 6-18, 6-19 mostram os dados utilizados para compor a Métrica M6 Quanto ao custo por hora de cada profissional envolvido na correção dos defeitos.

Quadro 6-16 – Resultado da M6: Custo da hora trabalhada de cada profissional envolvido diretamente na correção.

| **Cargos** | **Hora (custo(R$)) :** |
|---|---|
| **Analista de Requisitos** | 25,52 |
| **Desenvolvedor** | 19,89 |
| **Analista de Teste** | 19,89 |

Quadro 6-17 – Resultado da M6: Custo total de correção dos Defeitos encontrados.

| Tempo de Correção e Custo | | | |
|---|---|---|---|
| **Complexidade** | **Tempo médio (h)** | **Custo (R$)** | **Custo total por Bugs (R$)** |
| **Bug Simples** | 2 | 39.77 | 1.829,42 |
| **Bug Médio** | 4 | 90.81 | 15.438,45 |
| **Bug Complexo** | 8 | 181.63 | 3.995,83 |
| **Total** | | | 10.480,11 |

Quadro 6-18 – Resultado da M6: Custo totais do Teste para as entregas.

| Tempo de Teste e Custo | | | |
|---|---|---|---|
| **Complexidade** | **Tempo médio(h)** | **Número de U.C** | **Custo de Teste (R$)** |
| **U.C Simples** | 3 | 5 | 894,89 |
| **U.C Médio** | 5 | 2 | 994,32 |
| **U.C Complexo** | 12 | 3 | 8.590,91 |
| | | **TOTAL:** | 10.480,11 |

Quadro 6-19 – Resultado da M6: Aplicação da lei 10 de myers

| **Aplicando a Lei 10 de Myers** | **Custo Estimado Defeito** |
|---|---|
| **Teste** | 21.263,70 |
| **Homologação** | 212.637,00 |
| **Produção** | 2.126.370,03 |

O Quadro 6-20 mostram o resultado para a avaliação da aderência ao modelo de referencia TMM.

Quadro 6-20 – Resultado da M7: Resultado da avaliação de aderência ao TMM.

| TMM | | | | |
|---|---|---|---|---|
| **Níveis** | **Descrição dos objetivos do teste** | **Explicações** | **Fonte da evidência** | **Final** |
| **1** | **Teste normalmente feito pela equipe de desenvolvimento de forma rudimentar** | A atividade de testar é um processo caótico e não existe uma diferença entre testar e buscar erros. O teste é executado sem ferramentas, equipes treinadas e outros recursos. | ***Testes de Unidade*** | T |
| **2** | **Desenvolver os objetivos do teste** | O propósito do teste é mostrar que o software funciona. | ***Plano de Teste / SLA do projeto*** | T |

| TMM | | | | |
|---|---|---|---|---|
| **Níveis** | **Descrição dos objetivos do teste** | **Explicações** | **Fonte da evidência** | **Final** |
| | **Iniciar um processo de planejamento de teste** | | **Processo de Teste** | T |
| | **Institucionalizar técnicas e métodos básicos de teste** | | **Plano de Testes / CVS** | T |
| **3** | **Estabelecer uma organização de teste de software** | O propósito do teste é mostrar que o software não funciona | **Plano de Testes / Manual Operacional de Teste** | T |
| | **Integrar o teste no ciclo de vida do software** | | **Processo Geral de Desenvolvimento de Software / Cronograma de Projeto** | T |
| | **Controlar e monitorar o processo de teste** | | **Planlilha de Acompanhamento / Cronograma / Relatório de Teste** | T |
| | **Estabelecer um programa de treinamento** | | **WorkShop Interno / Manual Operacional de Teste** | T |
| **4** | **Estabelecer um programa amplo de revisão** | O propósito do teste não é provar nada mas reduzir os riscos causados pelos defeitos a níveis aceitáveis. | | NA |
| | **Estabelecer um programa de medições de teste** | | | NA |
| | **Evoluir a qualidade do software** | | | NA |
| **5** | **Aplicar processos de prevenção de defeitos** | A atividade de testar não é um ato, mas uma disciplina mental que resulta em baixos riscos para o software com pouco esforço de teste. | | NA |
| | **Controlar a qualidade** | | | NA |
| | **Otimizar o processo de teste** | | | NA |

# Apêndice C

# Questionário(*Survey*)

Abaixo é apresentado o questionário qualitativo aplicado na fábrica de software.

## Questionário

### Introdução

Com a evolução do processo de Teste de Software (TST) e Homologação (HOM), assim como com a evolução do processo de desenvolvimento de software da fábrica, percebeu-se: um aumento de produtividade da equipe alocada para o projeto; aumento de qualidade do produto; e também aumento da satisfação do cliente. Os motivos da satisfação devem-se: o cliente passou a acompanhar o andamento do projeto mais de perto; a transparência nos testes, já que toda documentação é disponibilizada para o mesmo; assim como a participação da equipe técnica do cliente na tomada de decisão das soluções do projeto; e a realização contínua de treinamentos ao cliente sistema levando em consideração o produto entregue. Com o desenvolvimento interativo, a partir dos módulos funcionais, o cliente consegue homologar o sistema com maior detalhe, garantindo maior precisão no atendimento das suas expectativas. Porém, esse processo está passando por constantes atualizações de forma que possa gerar um produto com melhor qualidade.

### Objetivo

O processos de TST e HOM necessitam de algumas alterações para se tornarem aderente às recomendações de um programa de melhoria da qualidade organizacional. Um dos focos é o modelo MPS.BR . Estas alterações ainda estão sendo diagnosticadas e em desenvolvimento para serem institucionalizadas em forma de procedimentos a serem incorporados ao processo.

O objetivo deste trabalho retrata a consolidação de um relato de experiência que descreve a evolução dos processos de TST e HOM, o qual se encontram em constante melhorias para compor os resultados parciais de estudos provenientes da aplicação de um trabalho de melhoria do processo organizacional no contexto de uma dissertação de mestrado do PPGCC/UFPA – Programa de Pós-Graduação em Ciência da Computação. Este resultado está alinhado com os interesses organizacionais na aplicação de uma avaliação MPS.BR a partir das boas práticas constantes no cenário dos processos de TST e HOM deste modelo.

### Instruções para responder as perguntas

Este questionário é dividido em duas seções. A primeira A.1 é relacionada ao Perfil do participante. Essas informações são úteis para saber qual a formação do participante no contexto de Teste de software.

A segunda, seção A.2, trata exclusivamente dos questionamentos referentes a parte do processo de software aplicado em uma Fábrica de Software, com foco nos ativos relacionados aos processos de TST e HOM. Para responder as questões é necessário um entendimento do processo de software em avaliação, descrito no **Anexo 1**.

**Exemplo de Preenchimento:**

| | |
|---|---|
| **Errado** | |
| **Instituição de origem?** | ( •) Pública<br>( ) Particular |
| **Correto** | |
| **Instituição de origem?** | (X) Pública<br>( ) Particular |

## A.1 Questionário do Perfil do Participante

| **Formação** | |
|---|---|
| **Instituição de origem?** | ( ) Pública<br>( ) Particular |
| **Tipo de curso?** | ( ) Engenharia<br>( ) Informática / Ciência da Computação<br>( ) Matemática<br>( ) Outro(s):_____ |
| **Grau de escolaridade?** | ( ) Ensino Médio Técnico<br>( ) Graduação<br>( ) Pós Graduação<br>( ) MBA |
| **Experiência** | |
| **Como você classificaria seu entendimento sobre teste de software?** | ( ) Excelente<br>( ) Alto<br>( ) Bom<br>( ) Médio<br>( ) Baixo<br>( ) Nenhum |
| **Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação?** | ( ) Analista de Teste<br>( ) Desenvolvedor<br>( ) Gerente de Processo<br>( ) Gerente de Projeto<br>( ) Coordenador de Teste de Software<br>( ) Analista de Requisitos<br>( ) SQA<br>( ) Usuário<br>( ) Analista de Homologação |
| **Já participou do desenvolvimento de que tipo de sistemas de computação?** | ( ) Grande Porte<br>( ) Médio Porte<br>( ) Pequeno Porte |
| **Em qual área da Engenharia de Software você mais atuou?** | ( ) Especificação de Requisitos<br>( ) Projeto<br>( ) Codificação |

| | |
|---|---|
| | ( ) Teste<br>( ) Manutenção<br>( ) Garantia / Controle da Qualidade<br>( ) Homologação |
| **Como você classificaria seu entendimento em Homologação?** | ( ) Excelente<br>( ) Alto<br>( ) Bom<br>( ) Médio<br>( ) Baixo<br>( ) Nenhum |
| **Já participou de quantos processos de software?** | ( ) Nenhum<br>( ) Entre 1 e 2<br>( ) Entre 3 e 7<br>( ) Acima de 7 |
| **Quantos processos de software já definiu?** | ( ) Nenhum<br>( ) Entre 1 e 2<br>( ) Entre 3 e 7<br>( ) Acima de 7 |
| **Como você classificaria seu entendimento sobre qualidade de software?** | ( ) Excelente<br>( ) Alto<br>( ) Bom<br>( ) Médio<br>( ) Baixo<br>( ) Nenhum |
| **Já usou quantos modelos/normas da qualidade?** | ( ) Nenhum<br>( ) Entre 1 e 2<br>( ) Entre 3 e 7<br>( ) Acima de 7 |

## A.2 Questionário de Definição do Processo

Nesta segunda etapa você irá avaliar o processo mostrado no Anexo 1. O objetivo deste questionário é validar a utilização deste processo no cenário de aplicação de uma fábrica de software.
Para realizar o preenchimento do questionário, selecione uma das colunas de acordo com as seguintes características:

- Presença: Se uma determinada atividade está presente no Processo de Desenvolvimento de Software apresentado, no contexto dos processos de TST e HOM.
- Utilidade: Se uma determinada atividade tem utilidade no processo de Desenvolvimento de Software, no contexto dos processos de TST e HOM.
- Adequação: Se uma determinada atividade necessita de alguma adequação para o processo de Desenvolvimento de Software, no contexto dos processos de TST e HOM.

Os critérios usados para a avaliação das características listadas são:

| Presença | Utilidade | Adequação do nível de detalhamento |
|---|---|---|
| 1. Não e´ oferecido pelo processo e não gostaria de tivesse disponível. | 1. Não e´ útil. | 1. O detalhamento deve ser aumentado. |
| 2. Não oferecido, mas gostaria que tivesse disponível. | 2. Provavelmente e´ útil, mas ainda não apliquei. | 2. O detalhamento não precisa ser modificado. |

| Presença | Utilidade | Adequação do nível de detalhamento |
|---|---|---|
| 3. Oferecido, parcialmente. | 3. E´ útil e já´ apliquei em diferentes implementações. | 3. O detalhamento deve ser diminuído. |
| 4. Oferecido. | | |

**Informações para o Preenchimento**
**Errado**

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | Desenvolvimento | Desenvolver Requisito | 1 | ● | 3 | 4 | 1 | 2 | ● | 4 | 1 | 2 | ● | 4 |

**Correto**

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | Desenvolvimento | Desenvolver Requisito | 1 | **X** | 3 | 4 | 1 | 2 | **X** | 4 | **X** | 2 | 3 | 4 |

## Atividades do Processo

| Nº | Equipe | Atividade | Presença | | | | Utilidade | | | Adequação | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Testes | Gerar Guia Operacional | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 2 | | Gerar Plano de Testes | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 3 | | Gerar Projeto de Testes | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 4 | | Gerar Script de Testes | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 5 | | Executar Script de Teste | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 6 | | Executar Projeto de Testes | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 7 | | Registrar Defeitos | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 8 | | Liberar Build | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 9 | Homologação | Homologar Sistemas | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 10 | | Registrar Problemas | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 11 | | Analisar Problemas | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 12 | | Comunicar Cliente | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 13 | | Gerar Plano de Homologação | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |

Para as atividades, que por um acaso não tenham sido encontrados na listagem acima, descreva no quadro a seguir todos estas atividades, e avalie as colunas correspondentes segundo as mesmas características e critérios anteriormente utilizadas como parâmetro quanto ao uso das atividades constantes no processo de Desenvolvimento de Software, no contexto dos processos de TST e HOM, listados por você no questionário:

| Nº | Equipe | Atividade | Presença | | | | Utilidade | | | Adequação | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 2 | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 3 | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 4 | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |
| 5 | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 |

# Anexo 1 do questionário

A Figura 1 exibe como está definido e institucionalizado atualmente o processo de Teste de Software utilizado na Fábrica de Software apresentada.

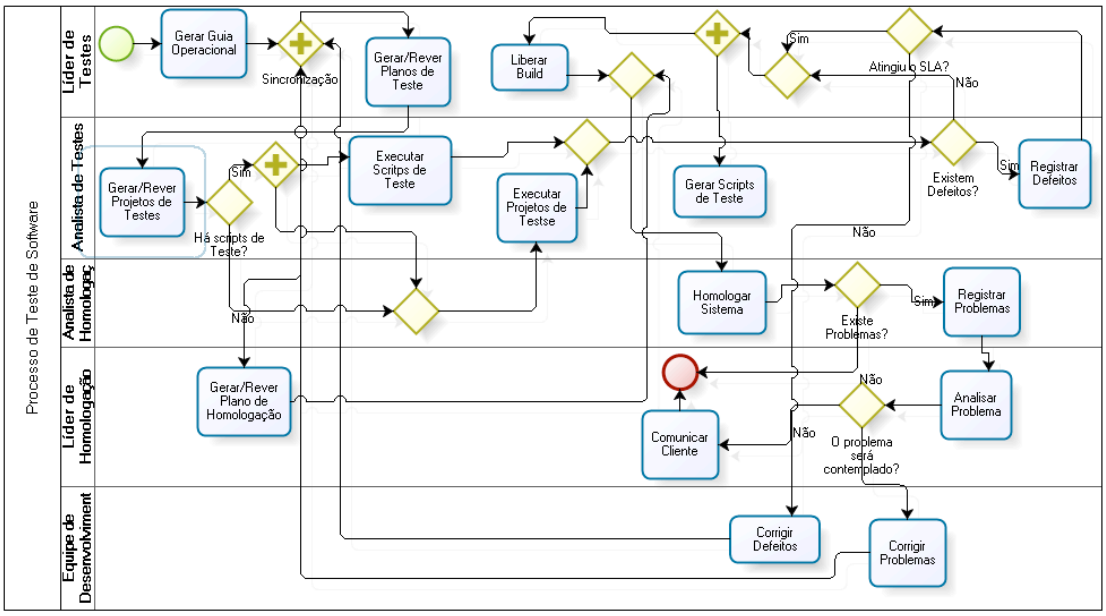


**Figura. 1**. Fluxo Detalhado do processo de teste de software executado na Fábrica de Software.

# Apêndice D

# Resultado do Questionário

Após a aplicação do questionário, o resultado coletado foi analisado conforme os Quadros abaixo.

Tabela 6-1– Resultado do Questionário: Tipo de Curso.

| **Tipo de curso?** | | | | |
|---|---|---|---|---|
| **Engenharia** | **Informática / Ciência da Computação** | **Matemática** | **Outro(s):** | **Total** |
| 0% | 100% | 0% | 0% | 100% |

Tabela 6-2 – Resultado do Questionário: Grau de Escolaridade.

| **Grau de escolaridade?** | | | | |
|---|---|---|---|---|
| **Ensino Médio Técnico** | **Graduação** | **Pós Graduação** | **MBA** | **Total** |
| 0% | 33% | 0% | 67% | 100% |

Tabela 6-3– Resultado do Questionário: Conhecimento em Teste de Software.

| **Como você classificaria seu entendimento sobre Teste de Software?** | | | | | | |
|---|---|---|---|---|---|---|
| Excelente | Alto | Bom | Médio | Baixo | Nenhum | Total |
| 16% | 50% | 17% | 17% | 0% | 0% | **100%** |

Tabela 6-4 – Resultado do Questionário: Experiência ou Cargo que mais exerceu.

| **Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação?** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Analista de Sistemas** | **Codificação** | **Gerente de Processo** | **Gerente de Projeto** | **Projetista de Processo** | **Projetista de Sistemas** | **SQA** | **Teste** | **Usuário** | **Total** |
| 0% | 0% | 0% | 0% | 0% | 0% | 0% | 100% | 0% | **100%** |

Tabela 6-5 – Resultado do Questionário: Já participou do desenvolvimento de que tipo de sistemas de computação.

| **Já participou do desenvolvimento de que tipo de sistemas de computação?** | | | |
|---|---|---|---|
| **Grande Porte** | **Médio Porte** | **Pequeno Porte** | **Total** |
| 100% | 0% | 0% | **100%** |

Tabela 6-6 – Resultado do Questionário: Área em que mais trabalhou.

| **Em qual área da Engenharia de Software você mais atuou?** | | | | | | |
|---|---|---|---|---|---|---|
| **Especificação de Requisitos** | **Projeto** | **Codificação** | **Teste** | **Manutenção** | **Garantia / Controle da Qualidade** | **Total** |
| 0% | 0% | 0% | 100% | 0% | 0% | **100%** |

Tabela 6-7 – Resultado do Questionário: Já participou de quanto processos de Software?

| **Já participou de quantos processos de software?** | | | | |
|---|---|---|---|---|
| **Nenhum** | **Entre 1 e 2** | **Entre 3 e 7** | **Acima de 7** | **Total** |
| 0% | 100% | 0% | 0% | **100%** |

Tabela 6-8 – Resultado do Questionário: Conhecimento em Homologação.

| **Como você classificaria seu entendimento sobre Homologação?** | | | | | | |
|---|---|---|---|---|---|---|
| Excelente | Alto | Bom | Médio | Baixo | Nenhum | Total |
| 17% | 0% | 83% | 0% | 0% | 0% | **100%** |

Tabela 6-9 – Resultado do Questionário: Conhecimento em Qualidade de Software.

| **Como você classificaria seu entendimento sobre qualidade de software?** |
|---|

| Excelente | Alto | Bom | Médio | Baixo | Nenhum | Total |
|---|---|---|---|---|---|---|
| 0% | 17% | 83% | 0% | 0% | 0% | **100%** |

Tabela 6-10 – Resultado do Questionário: Conhecimento em Qualidade de Software.

| **Já usou quantos modelos/normas da qualidade?** | | | | |
|---|---|---|---|---|
| **Nenhum** | **Entre 1 e 2** | **Entre 3 e 7** | **Acima de 7** | **Total** |
| 0% | 100% | 0% | 0% | **100%** |

Tabela 6-11 – Resultado do Questionário para as Atividades do Processo de Teste de Software.

| Nº | Equipe | Atividade | Presença | | | | Utilidade | | | Adequação | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | |
| 1 | Testes | Gerar Guia Operacional | | 16,66% | | 83.33% | | 66.66% | 33.33% | 33.33% | 66.66% | |
| 2 | | Gerar Plano de Testes | | 16.66% | | 83.33% | | 50% | 50% | 16.66% | 83.33% | |
| 3 | | Gerar Projeto de Testes | | 16.66% | | 83.33% | | | 100% | 33.33% | 50% | 16.66 |
| 4 | | Gerar Script de Testes | | 16.66% | 33.33% | 50% | | 50% | 50% | 33.33% | 66.66% | |
| 5 | | Executar Script de Teste | | 16.66% | 33.33% | 50% | | 50% | 50% | 33.33% | 66.66% | |
| 6 | | Executar Projeto de Testes | | | | 100% | | | 100% | 16.66% | 66.66% | 16.66 |
| 7 | | Registrar Defeitos | | | | 100% | | | 100% | | 83.33% | 16.66 |
| 8 | | Liberar Build | | | | 100% | | 16.66% | 83.33% | 33.33% | 66.66% | |
| 9 | Homologação | Homologar Sistemas | | | | 100% | | 50% | 50% | 33.33% | 50% | 16.66 |
| 10 | | Registrar Problemas | | | 16.66% | 83.33% | | 33.33% | 66.66% | 16.66% | 66.66% | 16.66 |
| 11 | | Analisar Problemas | | | 16.66% | 83.33% | | 50% | 50% | 16.66% | 66.66% | 16.66 |
| 12 | | Comunicar Cliente | | 16.33% | 16.33% | 66.66% | | 33.33% | 66.66% | 50% | 50% | |
| 13 | | Gerar Plano de Homologação | | | | 100% | | 66.66% | 33.33% | 33.33% | 66.66% | |

# Apêndice E

# Quadros de Atividades

Neste apêndice, são apresentados o detalhamento das atividades que fazem parte do processo de teste de software de versão 2. Os Quadros orientam quanto a Objetivo, critérios de Entrada, artefatos de saída, ações, critérios de saída, artefatos de saída, responsáveis, *templates*, métodos, ferramentas.

# Atividades

Gerar Guia Operacional de Testes

| Objetivo | |
|---|---|
| • O objetivo desta atividade é gerar um documento onde seja utilizado para descrever as atividades que serão realizadas no processo de testes e responsabilidades de cada participante do processo.<br>• O documento deve servir de base para os demais documentos relacionados da disciplina de Testes de Software. | |
| Critérios de Entrada | Artefatos de Entrada |
| • Processo de desenvolvimento de software em definição. | |
| Ações | |
| • Levantar técnicas e estratégias de teste de software que serão utilizadas no projeto.<br>• Descrever responsabilidades de cada papel participante.<br>• Estabelecer itens alvo de testes.<br>• Definir procedimentos de aplicação das técnicas e estratégias. | |
| Critérios de Saída | Artefatos de Saída |
| Guia Operacional de Teste pronto. | Guia Operacional de Teste |

| Responsáveis/ Papéis Envolvidos |
|---|
| Gerente de Projeto, Gerente de processo, Líder de Teste |
| **Templates** |
| • Template de Guia Operacional de Teste |
| **Métodos/Técnicas/Guias/Procedimentos** |
| • Teste Caixa Preta<br>  ○ Teste Funcional<br>  ○ Teste de Conformidade<br>  ○ Teste de Performance<br>  ○ Teste de Interface<br>  ○ Teste de Usabilidade |
| **Ferramentas de Apoio Utilizadas** |
| • Editor de Texto |

Gerar/Rever Plano de Teste

| Objetivo | |
|---|---|
| • O objetivo desta Atividade é gerar o plano de Teste.<br>• O plano de Teste possui como objetivo de elicitar as necessidades do teste do produto ou componente do produto.<br>• Estabelecer a estratégia de testes que será usada para o sistema ou parte em questão.<br>• Estabelecer responsáveis e cronograma. | |
| **Critérios de Entrada** | **Artefatos de Entrada** |
| • A Elicitação dos requisitos do sistema ou parte do sistema estejam estáveis e finalizados. | • Guia Operacional de teste;<br>• Documento de Requisitos;<br>• Documento de Regra de Negócios<br>• Dicionário de Dados; |
| **Ações** | |
| • Receber a formalização do Gerente do Projeto sobre demanda de execução dos testes<br>• Levantar técnicas e estratégias de teste de software que serão utilizadas no sistema ou módulo em questão.<br>• Estabelecer cronogramas do teste interno (equipe técnica de desenvolvimento).<br>• Identificar itens alvo de testes para o módulo ou sistema.<br>• Identificar e Gerenciar recursos necessários para os testes.<br>• Estabelecer critério de parada da atividade de teste.<br>• Estabelecer critérios de Cobertura da atividade de teste. | |
| **Critérios de Saída** | **Artefatos de Saída** |
| • Teste do módulo Planejado.<br>• Recursos comprometidos com os planejamentos. | • Plano de Teste |
| **Responsáveis/ Papéis Envolvidos** | |
| Líder de Teste e Analista de Testes | |
| **Templates** | |
| • Template de Plano de Teste. | |
| **Métodos/Técnicas/Guias/Procedimentos** | |
| • Cobertura de Pareto<br>• Cobertura dos caminhos críticos<br>• Cobertura de fluxos principais | |
| **Ferramentas de Apoio Utilizadas** | |
| • Editor de Texto.<br>• Editor de Cronogramas. | |

Gerar/Rever Plano de Homologação.

| Objetivo |
|---|
| • O objetivo desta Atividade é gerar o Documento de plano de Homologação.<br>• O plano de Homologação possui como objetivo guiar o cliente no processo de Homologação (aceitação) do produto ou parte do produto.<br>• O plano de Homologação possui também o objetivo de elencar necessidades e insumos |

| | |
|---|---|
| necessários para o bom funcionamento do sistema em ambiente de homologação. | |
| Critérios de Entrada | Artefatos de Entrada |
| • A Elicitação dos requisitos do sistema ou parte do sistema estejam estáveis e finalizados. | • Guia Operacional de teste;<br>• Plano de Teste. |
| Ações | |
| • Receber formalização do Gerente do Projeto sobre a demanda de homologação<br>• Estabelecer cronogramas de Homologação.<br>• Identificar itens alvo de testes para o produto ou parte do produto em homologação.<br>• Identificar e Gerenciar recursos (hardware, software, humano) necessários para a homologação.<br>• Identificar necessidades de infra-estutura para que o produto ou parte do produto possa ser homologado. | |
| Critérios de Saída | Artefatos de Saída |
| • Diretrizes do planejamento estarem definidas.<br>• Recursos comprometidos com o planejamentos. | • Plano de Homologação. |
| Responsáveis/ Papéis Envolvidos | |
| Líder de Teste e Analista de Testes | |
| Templates | |
| • Template de Plano de Homologação. | |
| Métodos/Técnicas/Guias/Procedimentos | |
| | |
| Ferramentas de Apoio Utilizadas | |
| • Editor de Texto. | |

Gerar/Rever Projeto de Teste

| | |
|---|---|
| Objetivo | |
| • O objetivo desta Atividade é gerar o Documento de Projeto de Teste.<br>• O objetivo do Projeto de Teste é descrever os procedimentos a serem realizados e resultados esperados do teste para o produto ou parte do produto. | |
| Critérios de Entrada | Artefatos de Entrada |
| • A Elicitação dos requisitos do sistema ou parte do sistema estejam estáveis e finalizados.<br>• Plano de Teste de software finalizado | • Guia Operacional de teste;<br>• Plano de Teste.<br>• Documento de Caso de Uso .<br>• Documento de Regra de Negócio. |
| Ações | |
| • Elaborar casos e cenários de teste.<br>• Descrever passos realizados e critérios de sucesso. | |
| Critérios de Saída | Artefatos de Saída |
| • Diretrizes para a execução de testes definido. | • Projeto de Teste |
| Responsáveis/ Papéis Envolvidos | |
| Analista de Testes. | |
| Templates | |
| • Template de Projeto de Teste. | |
| Métodos/Técnicas/Guias/Procedimentos | |
| • Definir projeto de teste baseados nos requisitos especificados. | |
| Ferramentas de Apoio Utilizadas | |
| • Editor de Texto. | |

Executar Projeto de Teste/Scripts de Teste

| | |
|---|---|
| Objetivo | |
| • O objetivo desta Atividade é a execução do teste propriamente dito, seja manual ou automático ou ambos. | |
| Critérios de Entrada | Artefatos de Entrada |
| • A Elicitação dos requisitos do sistema ou | • Guia Operacional de teste; |

| | |
|---|---|
| parte do sistema estejam estáveis e finalizados.<br>• Plano de Teste de software finalizado.<br>• Projeto de Teste de Software Finalizado.<br>• Sistema desenvolvido e disponibilizado a equipe de testes. | • Plano de Teste.<br>• Documento de Caso de Uso .<br>• Documento de Regra de Negócio.<br>• Projeto de Teste<br>• *Scripts* de Teste desenvolvidos. |
| **Ações** | |
| • Executar casos de teste.<br>o Entender os cenários de teste definidos no Projeto de Teste e as Regras de Negócio;<br>o Realizar os passos definidos no cenário de teste e nas regras de negócio usando como base o produto ou parte do produto;<br>o Para testes automatizados (performance e regressivo, por exemplo), analisa-se a definição dos requisitos funcionais e não-funcionais e *scripts* são elaborados para a automação.<br>• Registrar não conformidades no relatório. | |
| **Critérios de Saída** | **Artefatos de Saída** |
| • Execução completa do projeto de teste.<br>• Não conformidades corrigidas. | • Relatório de Teste. |
| **Responsáveis/ Papéis Envolvidos** | |
| Analista de Testes | |
| **Templates** | |
| • Template de Relatórios de Teste | |
| **Métodos/Técnicas/Guias/Procedimentos** | |
| • A cada ciclo de teste executado, mudar os componentes de ambientes usados<br>o Trocar de Navegador de Internet, Sistema Operacional e Capacidade computacional.<br>• Estabelecer critério de parada de acordo com o SLA acordado com o cliente no início do projeto. | |
| **Ferramentas de Apoio Utilizadas** | |
| • Editor de Texto.<br>• Navegadores.<br>• Ferramenta de automação de teste regressivo<br>• Ferramenta de *Bugtracking*<br>• Ferramenta de teste de performance. | |

Registrar Defeitos

| | |
|---|---|
| **Objetivo** | |
| • O objetivo desta Atividade é o registro de Defeitos encontradas no teste. | |
| **Critérios de Entrada** | **Artefatos de Entrada** |
| • Não conformidades Encontradas. | • Relatório de Teste |
| **Ações** | |
| • Registrar não conformidades em Ferramenta de *Bugtracking*.<br>o Preencher descrição do BUG.<br>o Indicar versão da ocorrência do BUG.<br>o Atribuir ao responsável pela correção.<br>• Acompanhar status do registro.<br>• Re-testar não conformidade após correção. | |
| **Critérios de Saída** | **Artefatos de Saída** |
| • Não conformidades registradas em ferramenta de bugtracking. | • Registro de não conformidades na ferramenta de *bugtracking*. |
| **Responsáveis/ Papéis Envolvidos** | |
| Analista de Testes | |
| **Templates** | |
| | |
| **Métodos/Técnicas/Guias/Procedimentos** | |
| • Registrar não conformidade na ferramenta de *Bugtracking*<br>• Registrar não conformidade em relatório de teste de software. | |
| **Ferramentas de Apoio Utilizadas** | |

| |
|---|
| • Editor de Texto.<br>• Ferramenta de *Bugtracking*. |

Gerar *Scripts* de Teste

| Objetivo | |
|---|---|
| • O objetivo desta Atividade é a Geração dos *scripts* de Testes Regressivos | |
| Critérios de Entrada | Artefatos de Entrada |
| • Produto ou sistema estável. | • Documento de Projeto de Testes |
| Ações | |
| • Gerar os *Scripts*.<br>o Através da ferramenta de captura de ações capturar os passos a serem percorridos pelo *scripts*<br>o Testar os *scripts*.<br>o Colocar os *scripts* na ferramenta de execução online. | |
| Critérios de Saída | Artefatos de Saída |
| • *Script* Executável. | • *Scripts* gerados<br>• *Scripts* disponíveis para execução. |
| Responsáveis/ Papéis Envolvidos | |
| Líder de Teste, Analista de Testes. | |
| Templates | |
| N/A | |
| Métodos/Técnicas/Guias/Procedimentos | |
| • Desenvolver os *scripts* de acordo com os casos de usos selecionados. | |
| Ferramentas de Apoio Utilizadas | |
| • Ferramenta de Geração de *Scripts*;<br>• Servidor de Aplicação JAVA; | |

Liberar *Build*

| Objetivo | |
|---|---|
| • O objetivo desta Atividade é realizar a liberação do produto ou parte do produto para homologação do cliente. | |
| Critérios de Entrada | Artefatos de Entrada |
| • Produto ou sistema testado. | • Documento Relatório de teste do módulo |
| Ações | |
| • Realizar entrega para Homologação.<br>o Criar uma TAG (rótulo que controla a versão em uma ferramenta automatizada) dos produtos de trabalho entregáveis (Documentação + Código).<br>o Realizar apresentação do produto entregue ao cliente.<br>• Colocar o produto ou parte do produto de maneira usável (atendendo a SLA's definidos pela equipe de teste, e acordados com o cliente) em um ambiente de homologação. | |
| Critérios de Saída | Artefatos de Saída |
| • Aprovação do Líder de Teste. | • Documento de Entrega.<br>• Resultados de Testes. |
| Responsáveis/ Papéis Envolvidos | |
| Líder de Teste | |
| Templates | |
| Template de Termo de entrega. | |
| Métodos/Técnicas/Guias/Procedimentos | |
| • Gerar *Baseline* de entrega.<br>• Promover o sistema para ambiente de homologação.<br>• Verificar *checklist* de entregáveis. | |
| Ferramentas de Apoio Utilizadas | |
| • Editor de Texto<br>• Ferramenta de Gerência de Configuração. | |

Homologar Sistema

| Objetivo | |
|---|---|
| • O objetivo desta Atividade é realizar a homologação do produto ou parte do produto com o Cliente. | |
| Critérios de Entrada | Artefatos de Entrada |
| • Produto ou sistema testado. | • Requisitos;<br>• Projeto de Teste;<br>• Casos de Usos. |
| Ações | |
| • Realizar apresentação do sistema para o cliente.<br>• Executar cada requisito ou caso de uso em homologação juntamente com o cliente;<br>• . | |
| Critérios de Saída | Artefatos de Saída |
| • Homologação de sistema; | • Carta de Homologação.<br>• Registro de Defeitos. |
| Responsáveis/ Papéis Envolvidos | |
| Analista de Homologação | |
| Templates | |
| Template de Termo de Homologação. | |
| Métodos/Técnicas/Guias/Procedimentos | |
| • Análise de defeitos caso haja;<br>• Explicação do Negócio envolvido; | |
| Ferramentas de Apoio Utilizadas | |
| • Editor de Texto;<br>• Navegador;<br>• Sistema entregue. | |

Analisar Problema

| Objetivo | |
|---|---|
| • O objetivo desta Atividade Analisar os problemas relatados em fase de homologação afim de checar se está classificado corretamente quanto sua criticidade e se o problema realmente persiste.. | |
| Critérios de Entrada | Artefatos de Entrada |
| • Problema Registrado pelo Cliente. | • Registro do problema. |
| Ações | |
| • Realizar análise do problema.<br>• Reclassificar criticidade do problema caso haja necessidade;<br>• Encaminhar problema para o responsável pela correção. | |
| Critérios de Saída | Artefatos de Saída |
| • Problema analisado e encaminhado; | • Registro encaminhado. |
| Responsáveis/ Papéis Envolvidos | |
| Analista de Homologação;<br>Cliente. | |
| Templates | |
| Registro de Problemas. | |
| Métodos/Técnicas/Guias/Procedimentos | |
| • Analise da criticidade de acordo com o SLA estabelecido; | |
| Ferramentas de Apoio Utilizadas | |
| • Ferramenta de *Bugtracking*;<br>• Sistema entregue. | |