GLASIANA APARECIDA CRUZ

AVALIAÇÃO DE MATURIDADE EM PROCESSO DE TESTE DE SOFTWARE: UMA PESQUISA-AÇÃO

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

LAVRAS
MINAS GERAIS - BRASIL
2010

GLASIANA APARECIDA CRUZ

AVALIAÇÃO DE MATURIDADE EM PROCESSO DE TESTE DE SOFTWARE: UMA PESQUISA-AÇÃO

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

Área de Concentração:
Engenharia de Software, Teste de Software

Orientador:
Dr. Paulo Henrique de Souza Bermejo

LAVRAS
MINAS GERAIS - BRASIL
2010

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

Cruz, Glasiana Aparecida

Avaliação de Maturidade em Processo de Teste de Software: uma Pesquisa-Ação/ Glasiana Aparecida Cruz. Lavras – Minas Gerais, 2010.

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

1. Engenharia de Testes. 2. Engenharia de Software. I. CRUZ, G. A. II. Universidade Federal de Lavras. III. Título.

GLASIANA APARECIDA CRUZ

AVALIAÇÃO DE MATURIDADE EM PROCESSO DE TESTE DE SOFTWARE: UMA PESQUISA-AÇÃO

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

Aprovada em 17 de novembro de 2010

_____
Dr. Antônio Maria Pereira de Resende UFLA

_____
MSc. Geovane Nogueira Lima FAGAMMON

_____
Dr. Paulo Henrique de Souza Bermejo
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL
2010

“Porque, onde estiver o vosso tesouro, ali estará também o vosso coração.”
(Jesus Cristo)

Dedico esse trabalho aos meus pais Tadeu e Vicentina, meus irmãos, mais que irmãos Mônica e Carlinhos, meus cunhados Geraldo e Nicinha e meus lindos sobrinhos Nícolas e Tobias, que sempre me mostram o que realmente importa na vida e sempre me dão razão pra voltar pra casa e me dão motivos pra viver.
Dedico também a minha Tia Ana que me mostrou que existia um mundo inteiro lá fora, e que eu podia fazer parte dele também! Tudo começou com a senhora!

AGRADECIMENTOS

Agradeço a todos que de alguma forma fizeram parte dessa história.
Agradeço ao meu orientador Paulo Bermejo por não ter desistido de mim e por ter me ajudado a finalizar o trabalho.
Agradeço a minha família em Lavras, Aline e Flávia (Flora) que são companheiras para todos os momentos e me ajudaram muito em todos os momentos de sufoco.
Agradeço aos amigos que parecem que ainda moram comigo, que estão sempre presentes quando preciso: Júlio, Flávia e Straus, amizade é isso tudo que a gente tem!
Agradeço aos amigos Adler e Geovane, que sempre me ajudavam com conhecimento, quando eu precisava.
Aos amigos e companheiros da SWFactory, que me ensinaram que trabalhar também pode ser divertido.
Agradeço a tantas garotas que passaram pelo 212 lá no brejão, nesse tempo todo, e que fizeram participações mais que especiais na minha vida: Wanda, Renata, Kelly, Grazi, Dani agora mamãe da Júlia, Mariana, Parao, Lala (em especial), Pops, Lili, Lu, Fabiana e os agregados.
Aos cunhados Breno, Guilherme, amigos que animavam o dia a dia.
Aos amigos importantes: Eleuza, Leide, Sheila, Luci, Edvane, Rodolfo, Nando, Tássia e a turma melhor do univelso.
Aos amigos músicos que são os melhores que eu conheço e que me proporcionaram tantos momentos felizes, fazendo aquele som: Os internautas, Nélio, Jean, Maria, Diogo.
Agradeço a todos os amigos que fiz e desfiz e que sempre serão importantes na minha vida!

AVALIAÇÃO DE MATURIDADE EM PROCESSO DE TESTE DE SOFTWARE: UMA PESQUISA-AÇÃO

RESUMO

A busca por qualidade tem feito as empresas de TI investirem em testes, que representa uma etapa importante nessa busca, e com isso a área de teste de software passou a ter uma importância muito maior no mercado. Este trabalho apresenta uma avaliação do processo de teste de software de uma organização, analisando dois projetos, comparando as práticas dos projetos com as práticas do processo Gerência de Projetos de Teste de Software do nível 1 do modelo MPT.BR. Os resultados mostraram que a empresa possui boas práticas em teste de software, mas não é totalmente aderente ao nível. Algumas práticas importantes não são adotadas, e seu processo pode ser melhorado consideravelmente com a adoção dessas práticas. Algumas sugestões de melhorias foram propostas.

Palavras-chave: teste de software, MPT.BR, melhoria de processo de teste.

ASSESSMENT OF THE MATURITY IN PROCESS OS SOFTWARE TESTING: AN ACTION RESEARCH

ABSTRACT

The search for quality has made IT enterprises invest increasingly in software testing – an important stage to reach this goal. With that, the importance of testing area has grown expressively worldwide. This work presents an evaluation of software testing's process in an organization, analyzing two development projects, and comparing the practices applied to them with the practices of Software Testing Project Management level 1 of MPT.BR model. The results show that the organization has good practices for software testing, but it's not totally adherent to the verified level. The introduction of important practices that aren't adopted can be improved considerably the development process. Some suggestions of improvement are proposed at the end of this work.

Keywords: software testing, MPT.BR, software test process improvement

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e Motivação

Segundo Sommerville (2003) o termo software ainda é muito associado aos programas de computador, porém, essa visão é na verdade muito restrita. Além do programa, software também é toda documentação associada e os dados de configuração necessários para fazer com que esses programas operem corretamente.

Com o desenvolvimento cada vez maior da tecnologia de hardware e a consequente disponibilidade de máquinas cada vez mais potentes e baratas, o uso de computadores tem-se tornado cada vez mais difundido em diversas áreas. Isso tem feito com que aumente a demanda por software cada vez maior e mais complexo. (VASCONCELOS *et* 2006).

O cliente espera um produto que funcione de acordo com suas necessidades e para isso, ele passa todas as informações necessárias para que os programadores possam começar a fabricação do software, e quando ele recebe o produto já pronto para uso ele testa e espera que não ache erros e que ele funcione corretamente.

O custo da falta de qualidade pode ser muito significativo nos gastos de um projeto de software. Uma empresa que comercializa um produto busca o lucro, porém produzir um produto sem qualidade, nunca chamará atenção dos clientes que necessitam dele. E segundo Gomes (2009), pouco se fala a respeito dos custos resultantes dos defeitos ou erros provocados por falha de softwares, tanto para produtores quanto para usuários.

Hoje, a concorrência força todos a procurarem meios de melhorar a sua produção e se estabilizar no mercado, e com isso vem crescendo a preocupação com qualidade na área de desenvolvimento de software. Mas não somente um produto com qualidade é o que o mercado deseja, mas também um produto útil, fácil, objetivo, termos esses muito usados em Engenharia de Software, um dos alvos de nosso estudo.

E o principal alvo, o Teste de Software auxilia no aumento da qualidade de um produto que se encaixa nas necessidades do cliente. Melhorar o processo de testes implica em melhorar a qualidade e confiança dos produtos.

1.2 Objetivo e Estrutura do Trabalho

1.2.1 Objetivo geral

Este trabalho tem por objetivo geral avaliar o processo de teste realizado em projetos de uma organização com base em um modelo de maturidade de teste de software e propor melhorias ao processo.

1.2.2 Objetivos específicos

Para alcançar o objetivo geral deste trabalho, serão considerados os seguintes objetivos específicos:

1. Selecionar modelo de maturidade para apoiar a avaliação;
2. Diagnosticar como são realizadas atividades de teste em projetos de desenvolvimento de software da organização estudada;
3. Propor melhorias para elevar o nível de maturidade sobre o desenvolvimento e gerenciamento de atividades de teste.

1.2.3 Estrutura do trabalho

Este trabalho foi organizado da seguinte forma:

- O capítulo 2 abrange os conceitos relacionados a Teste de Software, importantes para se ter o entendimento sobre o estudo. A sessão 2.3 é dedicada aos conceitos sobre Modelos de Maturidade, apresentando alguns modelos, detalhando o MTP.BR, o modelo escolhido para avaliação.
- O capítulo 3 descreve a metodologia utilizada para se realizar o estudo.
- O capítulo 4 apresenta os resultados e discussões. Primeiramente na sessão 4.1 a organização é descrita, sem identificá-la. Na sessão 4.2 é descrito o processo de teste utilizado na organização. Na sessão 4.3 é realizada a discussão sobre a escolha do modelo selecionado para realizar a avaliação. Na sessão 4.4 os dois projetos escolhidos são brevemente descritos. Na sessão 4.5 os procedimentos técnicos necessários para realizar a avaliação

são apresentados. Na sessão 4.6 os resultados da avaliação são discutidos e na sessão 4.7 são sugeridas propostas de melhoria de acordo com os resultados encontrados.

- O capítulo 5 apresenta as conclusões, contribuições e possíveis trabalhos futuros a serem desenvolvidos a partir deste.

2 REFERENCIAL TEÓRICO

2.1 APRESENTAÇÃO

O presente capítulo trata dos principais conceitos de Engenharia de Software, e introduz o assunto Teste de Software no contexto do processo do software.

Ainda nesse capítulo é apresentada uma visão geral sobre modelos de maturidade de teste de software, abordando alguns modelos. Entre esses modelos, uma análise mais profunda é realizada a respeito do MPT.BR, o modelo escolhido para realizar a avaliação.

2.2 ENGENHARIA DE SOFTWARE

Segundo Sommerville (2006), Engenharia de Software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Nessa definição, existem dois termos importantes que devem ser compreendidos:

1. 'Disciplina de engenharia': aplica as teorias, métodos e ferramentas nas situações apropriadas, de modo seletivo; e sempre procura descobrir soluções para os problemas, mesmo quando não existem teorias aplicáveis e métodos de apoio.
2. 'Todos os aspectos da produção de software': A engenharia não se dedica exclusivamente aos processos técnicos de desenvolvimento de software, mas também a atividades como o gerenciamento de projetos de software e a desenvolvimento de ferramentas, métodos e teorias que dêem apoio à produção de software

Para Pressman (2006) o software é formado por programas que executam em computadores de qualquer tamanho e arquitetura, conteúdo que é apresentado ao programa a ser executado e documentos tanto em forma impressa quanto virtual que combinam todas

as formas de mídia eletrônica.

No passado, segundo Sommerville (2006), o desenvolvimento de software estava em crise. Projetos importantes sofriam atrasos, às vezes, de alguns anos. Por apresentarem custos muito maiores do que os inicialmente previstos, eles não eram confiáveis, eram de difícil manutenção e tinham desempenho inferior. Com o reconhecimento da existência da crise, foi proposto que o desenvolvimento de software deixasse de ser puramente artesanal e passasse a ser baseado em princípios de Engenharia (VASCONCELOS *et al*, 2006).

2.2.1 O Processo de Desenvolvimento de Software

Um processo de software essencialmente descreve a forma como uma organização desenvolve seus produtos de software e serviços de suporte, tais como documentação. Processos definem que medidas as organizações de desenvolvimento devem ter em cada fase da produção e prestam assistência em fazer estimativas, planos de desenvolvimentos, e medidas a qualidade (COLEMAN E O'CONNOR, 2007).

Um processo de software consiste em atividades, métodos e práticas úteis no desenvolvimento de um produto de software. As atividades associadas a esse processo de software incluem a descrição e preparação de esquemas que identifiquem a estrutura dos dados e elementos de controle de um sistema, codificação, verificação e implantação do software (PETERS & PEDRYCS, 2001).

Para Vasconcelos *et al* (2006) vários modelos de ciclo de vida têm sido propostos a fim de facilitar o entendimento do processo de desenvolvimento de software. Sommerville (2006) define as seguintes atividades fundamentais do ciclo de vida comuns a todos os processos, embora existam muitos diferentes:

- Especificação de software: É preciso definir a funcionalidade do software e as restrições em sua operação.
- Projeto e implementação do software: Deve ser produzido o software de modo que cumpra sua especificação.
- Validação de software: O software precisa ser validado para garantir que ele faz o que o cliente deseja.
- Evolução de software: O software precisa evoluir para atender às necessidades mutáveis do cliente.

2.2.2 Gestão de Qualidade de Software

Nos últimos anos, a qualidade do produto e serviço se tornou um diferencial para muitas empresas se distinguirem de suas concorrentes. Existem muitas definições de qualidade, Bastos *et al*. (2007) a define em dois pontos de vista: do ponto de vista do produtor, a qualidade é cumprimento de requisitos e do ponto de vista do consumidor é o atendimento às necessidades do cliente.

Sommerville (2006) estrutura a gestão de qualidade de software para sistemas de grande porte em três grandes atividades: garantia da qualidade que estabelece um framework de procedimentos organizacionais e normas que levam ao software de alta qualidade; planejamento da qualidade que seleciona procedimentos e normas adequado deste framework adaptado para um projeto de software específico e controle de qualidade que define e implementa os processos que garantem que a equipe de desenvolvimento seguiu os procedimentos e normas de qualidade.

É comum haver uma confusão na indústria a respeito da diferença entre controle de qualidade e garantia de qualidade. Bastos *et al*. (2007) mostra a diferença definindo as duas práticas a seguir:

Garantia de qualidade: A garantia da qualidade é um conjunto sistemático e planejado de atividades, necessárias para proporcionar a confiança adequada de que produtos e serviços estarão em conformidade com requisitos especificados e atenderão às necessidades do usuário. A garantia de qualidade deve ser uma função do pessoal da empresa responsável pela implementação da política de qualidade, definida através do desenvolvimento e da melhoria contínua do processo de desenvolvimento de software.

Controle de qualidade: O controle de qualidade é um processo pelo qual a qualidade do produto é comparada com os padrões aplicáveis; quando uma não-conformidade é detectada, são tomadas as devidas providências. O controle de qualidade é uma função em linha, e o trabalho é realizado num processo que assegura que o produto funcione conforme o padrão e os requisitos.

2.2.3 Normas e Modelos de maturidade em melhoria de processo de software

Diversos modelos, normas e padrões de qualidade de software vêm sendo propostos ao longo dos últimos anos. Esses modelos têm sido fortemente adotados por organizações

em todo o mundo (VASCONCELOS *et al*. 2006). Com essa padronização, segundo Peters e Pedrycz (2001), tornou-se possível medir tamanho, conteúdo, valor ou qualidade de uma entidade de software (PETERS & PEDRYCS, 2001).

As normas podem ser internacionais, regionais, nacionais e organizacionais em função da sua área de aplicação. Normas nacionais são editadas por uma organização nacional de normas. No Brasil, esta organização é a Associação Brasileira de Normas Técnicas - ABNT, e os organismos internacionais mais importantes para o setor de software são a ISO - International Organization for Standardization e a IEC - International Electrotechnical Commission (Villas Boas 2007).

Como exemplos de normas e modelos pode-se citar: ISO 9000, CMMI, ISO15504, ISO IEC, dentre outros. Será descrito a seguir, uma visão geral sobre os modelos CMMI e o MPS.BR.

2.2.3.1 CMMI – Modelo Integrado de Maturidade e Capacidade

Em 1987, o Software Engineering Institute – SEI publicou o primeiro modelo de processo criado para ser aplicado ao desenvolvimento de software: o SW-CMM. Ao logo do tempo foram publicados diversos outros modelos com finalidades específicas como engenharia de sistemas, aquisição, entre outros. Tais modelos apresentaram algumas inconsistências quando eram aplicados simultaneamente em uma mesma organização. Em março de 2002 foi publicado o CMMI, Modelo Integrado de Maturidade da Capacidade de Processo de Software cuja finalidade foi integrar todos os modelos em uma estrutura coerente e alinhada. (PESSÔA, 2008).

O modelo é estruturado em níveis referentes aos estágios de maturidade que a organização possui para desenvolver o software. Um nível de capacidade consiste em relacionar práticas genéricas e específicas para uma área de processo que pode melhorar os processos da organização associados com aquela área de processo. A área de processo é um conjunto de práticas relacionadas em uma área que, quando executadas coletivamente, satisfazem um conjunto de metas consideradas importantes para se ter uma melhora significativa nessa área. (DELGADO, 2007).

São cinco níveis de maturidade: Inicial, Gerenciado, Definido, Gerenciado Quantitativamente e Otimização. Estes níveis são apresentados na Tabela 1:

Tabela 1- Áreas de processo da representação por estágio do CMMI (Fonte: CMMI-DEV (2006))

Nível 2 – Gerenciado	Monitoramento e Controle de Projeto
	Planejamento de Projeto
	Garantia da Qualidade de Processo e Produto
	Medição e Análise
	Gestão de Configuração
	Gestão de Requisitos
	Gestão de Contrato com Fornecedores
Nível 3 – Definido	Análise e Tomada de Decisões
	Gestão Integrada de Projeto
	Definição dos Processos da Organização
	Foco nos Processos da Organização
	Treinamento na Organização
	Integração de Produto
	Desenvolvimento de Requisitos
	Gestão de Riscos
	Solução Técnica
	Validação
	Verificação
	Gestão Quantitativa de Projeto
Nível 4 – Gerenciado Quantitativamente	Desempenho dos Processos da Organização
Nível 5 – Em otimização	Implantação de Inovações na Organização
	Análise e Resolução de Causas

Os níveis de capacidade são definidos por Pessôa (2008) da seguinte forma:

- Nível 1 : Inicial – nesse nível, as organizações podem perfeitamente desenvolver produtos de software de alta qualidade, mas seu desempenho depende da competência das pessoas. No nível 1 não existe nenhuma área de processo;
- Nível 2: Gerenciado - os métodos de gerenciamento de software são

documentados e acompanhados nesse nível. Políticas organizacionais orientam os projetos estabelecendo processos de gerenciamento. Práticas bem sucedidas podem ser repetidas em novos projetos. No nível 2, são ao todo 7 áreas de processo: Monitoramento e Controle de Projeto, Planejamento de Projeto, Garantia da Qualidade de Processo e Produto, Medição e Análise, Gestão de Configuração, Gestão de Requisitos e Gestão de Contrato com Fornecedores;

- Nível 3: Definido – a organização possui um processo de engenharia bem definido. Existe uma preocupação com um processo padronizado para a organização, customizado para cada projeto. O projeto é definido, documentado e compreendido. São ao todo 12 áreas de processo: Análise e Tomada de Decisões, Gestão Integrada de Projeto, Definição dos Processos da Organização, Foco nos Processos da Organização, Treinamento na Organização, Integração de Produto, Desenvolvimento de Requisitos, Gestão de Riscos, Solução Técnica, Validação, Verificação, Gestão Quantitativa de Projeto;
- Nível 4: Gerenciado Quantitativamente – nesse nível a gerência tem bases objetivas para a tomada de decisão, pois o processo é medido e gerenciado quantitativamente. É possível prever o desempenho dentro de limites quantificados. O nível possui 1 área de processo: Desempenho dos Processos da Organização;
- Nível 5: Otimização – nesse nível o foco é na melhoria contínua do processo, onde a mudança de tecnologia e as mudanças no próprio processo são gerenciadas de forma a não causarem impacto na qualidade do produto final. No nível 5 são ao todo duas áreas de processo : Implantação de Inovações na Organização, Análise e Resolução de Causas.

2.2.3.2 MPS – Melhoria de Processo de Software

O programa MPS.BR (Melhoria de Processo do Software Brasileiro) foi criado como uma iniciativa para melhorar a capacidade de desenvolvimento de software nas organizações brasileiras. Esta iniciativa teve início em 2003 sob a coordenação da

Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), com apoio do governo (MCT e FINEP), SEBRAE/PROIMPE e BID (Banco Interamericano de Desenvolvimento), da indústria de software brasileira e de instituições de pesquisa. O principal objetivo desta iniciativa é desenvolver e disseminar um modelo de melhoria de processos brasileiro (o modelo de referência MPS) visando estabelecer um caminho economicamente viável para que organizações, incluindo as PMEs, alcancem os benefícios da melhoria de processos e da utilização de boas práticas da engenharia de software em um intervalo de tempo razoável. O modelo foi desenvolvido levando em consideração normas internacionais, modelos internacionalmente reconhecidos, boas práticas da engenharia de software e as necessidades de negócio da indústria de software brasileira. (KALINOWSKI *et al*, 2010).

O Projeto MPS-BR visa a melhoria de processo do software brasileiro, a um custo acessível, e segundo Weber *et al*. (2005) compreende duas metas:

- Desenvolvimento e aprimoramento do Modelo MPS, compatível com o CMMI e em conformidade com as normas ISO/IEC 12207 e ISO/IEC 15504;
- A implantação e avaliação do Modelo MPS, a um custo acessível em todas as regiões do país, com foco em grupos de pequenas e médias empresas (PMES)

Segundo SOFTEX (2005), o Modelo de Referência MPS define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado), com a escala de maturidade começando no nível G e terminando no nível A. Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. O nível de maturidade em que se encontra uma organização permite prever o seu desempenho futuro ao executar um ou mais processos.

A Tabela 2 mostra os níveis de maturidade com seus respectivos processos:

Tabela 2 - Níveis de maturidade e processos MPS.BR (Fonte: Koscianski; Soares (2007, p.145))

A	Inovação e implantação na organização
	Análise de causas e resolução
B	Desempenho do processo organizacional
	Gerência quantitativa do projeto
C	Análise de decisão e resolução
	Gerência de riscos

D	Desenvolvimento de requisitos
	Solução técnica
	Integração do produto
	Liberação do produto
	Instalação do produto
	Verificação
	Validação
E	Treinamento
	Avaliação e melhoria do processo organizacional
	Definição do processo organizacional
	Adaptação do processo para gerência de projeto
F	Medição
	Gerência de Configuração
	Aquisição
	Garantia da qualidade
G	Gerência de requisitos
	Gerência de projetos

A seguir, os níveis de maturidade são definidos por SOFTEX (2009a), de forma crescente são definidos:

- Nível G: Parcialmente Gerenciado - é composto pelos processos Gerência de Projetos e Gerência de Requisitos. Neste nível o processo deve ser executado e gerenciado;
- Nível F: Gerenciado - é composto pelos processos do nível de maturidade anterior (G) acrescidos dos processos Aquisição, Garantia da Qualidade, Gerência de Configuração, Gerência de Portfólio de Projetos e Medição. Neste nível o processo deve ser executado, gerenciado, e os produtos do trabalho devem ser gerenciados;
- Nível E: Parcialmente Definido - é composto pelos processos dos níveis de maturidade anteriores (G e F), acrescidos dos processos Avaliação e Melhoria do Processo Organizacional, Definição do Processo Organizacional, Gerência de Recursos Humanos e Gerência de Reutilização. O processo Gerência de Projetos sofre sua primeira evolução, retratando seu novo propósito: gerenciar o projeto com base no processo definido para o projeto e nos planos integrados. Neste nível o processo deve ser executado, gerenciado, os produtos do trabalho devem ser gerenciados, definido, e estar

implementado;

- Nível D: Largamente Definido - é composto pelos processos dos níveis de maturidade anteriores (G ao E), acrescidos dos processos Desenvolvimento de Requisitos, Integração do Produto, Projeto e Construção do Produto, Validação, e Verificação. Neste nível o processo deve ser executado, gerenciado, os produtos do trabalho devem ser gerenciados, definido, e estar implementado;
- Nível C: Definido - é composto pelos processos dos níveis de maturidade anteriores (G ao D), acrescidos dos processos Desenvolvimento para Reutilização, Gerência de Decisões e Gerência de Riscos. Neste nível o processo deve ser executado, gerenciado, os produtos do trabalho devem ser gerenciados, definido, e estar implementado;
- Nível B: Gerenciado Quantitativamente - é composto pelos processos dos níveis de maturidade anteriores (G ao C). Neste nível o processo de Gerência de Projetos sofre sua segunda evolução, sendo acrescentados novos resultados para atender aos objetivos de gerenciamento quantitativo. Neste nível o processo deve ser executado, gerenciado, os produtos do trabalho devem ser gerenciados, definido, estar implementado, parcialmente medido e parcialmente controlado;
- Nível A: Em Otimização - é composto pelos processos dos níveis de maturidade anteriores (G ao B). Neste nível o processo deve ser executado, gerenciado, os produtos do trabalho devem ser gerenciados, definido, estar implementado, parcialmente medido, parcialmente controlado. O processo é objeto de melhorias e inovações e é otimizado continuamente.

2.2 TESTES DE SOFTWARE

O Objetivo desse capítulo é mostrar uma visão geral sobre os testes, trazendo informações conceituais, classificando e mostrando a sua importância. Também serão abordados outros temas como o processo de teste, a documentação gerada durante o processo e as ferramentas de apoio ao processo.

2.2.1 Visão Geral

Vemos na literatura que o teste de software constitui em uma atividade no desenvolvimento do software, que segundo Vasconcelos *et al*. (2006) deve ser uma atividade cuidadosa e bem planejada, como parte dos esforços, no sentido de garantir a qualidade do software

É importante a identificação de alguns termos para uma melhor compreensão do assunto. A seguir, algumas definições, de acordo com Magela (2006):

- Erro: Resultado inconsistente na execução do software devido a uma ou mais falhas. Visão de verificação.
- Falha: Violação da especificação presente no código-fonte de um componente.
- Defeito: Conjunto de falhas que, quando provoca um comportamento inconsistente do software. Visão de validação.

As atividades de Verificação e Validação servem para assegurar que o software funcione de acordo com o que foi especificado e atenda aos requisitos dos stakeholders. Essas atividades constituem um processo iniciado com as revisões dos requisitos, passando pelas revisões da análise e do projeto do software e as inspeções do código até chegar aos testes (SOMMERVILLE, 2003).

Koscianski *et al*. (2007) admite que a verificação consiste em identificar defeitos e possíveis problemas de um componente pronto, enquanto a validação busca avaliar se a construção do componente segue os requisitos predefinidos.

E Magela (2006) complementa que o software poderá estar com erro, mas não com defeito – Um erro está relacionado ao lado formal da especificação e poderá ser imperceptível para o usuário final. Contudo, isso não significa que o software não esteja com erro. Assim, tem-se uma implicação matemática, ou seja, se tem um defeito, tem um erro, mas não o inverso.

2.2.2 O teste de software

Segundo Herbert (1999), teste de software constitui-se em um conjunto de atividades de validação e verificação com o objetivo de encontrar defeitos no produto e no

processo de software.

Já Myers (2004), define teste como sendo o ato de executar um programa fornecendo dados de entrada, verificando os resultados obtidos e comparando-os com os resultados esperados e com o objetivo de revelar a presença de defeitos, ou erros.

Somente após a realização da Primeira Conferência Formal em Teste de Software (na Universidade da Carolina do Norte, 1972) que a disciplina de teste ganhou caráter formal e várias outras conferências foram realizadas, assim como vários cientistas começaram a se dedicar a esse assunto estando hoje no contexto do processo de desenvolvimento de software (VILLAS BOAS, 2003).

De acordo com Swebok (2004) "O teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão da especificação, do projeto e da codificação". Por isso, o processo de teste deve ser executado durante todo o ciclo de vida do projeto do software.

O Instituto de Engenharia de Software (SEI), órgão criado pelo Departamento de Defesa americano estima que cerca de 55% de todo trabalho realizado pela área de desenvolvimento de software de uma empresa é destinado à correção de erros detectados em sistemas já em funcionamento.

Quanto mais cedo no processo de desenvolvimento de um software um ***bug*** for encontrado, menor será o custo de sua correção. Segundo Rex Black (2000), *bugs* encontrados pelos programadores custa R$ 10 para consertar e *bugs* encontrados pelos clientes custam US $ 1.000 para consertar.

Pressman (2006) complementa relatando que não é raro uma organização de desenvolvimento de software gastar entre 30 e 40% do esforço total do projeto de teste. A rigor, o teste de software que envolve vidas (por exemplo controle de vôo, monitoramento de reatores nucleares) pode custar de três a cinco vezes mais do que todos os outros passos de engenharia de software combinados!

Myers (2004) afirma que o custo de correção de *bugs* aumenta em 10 vezes a cada fase do desenvolvimento do software, conforme a Figura 1.

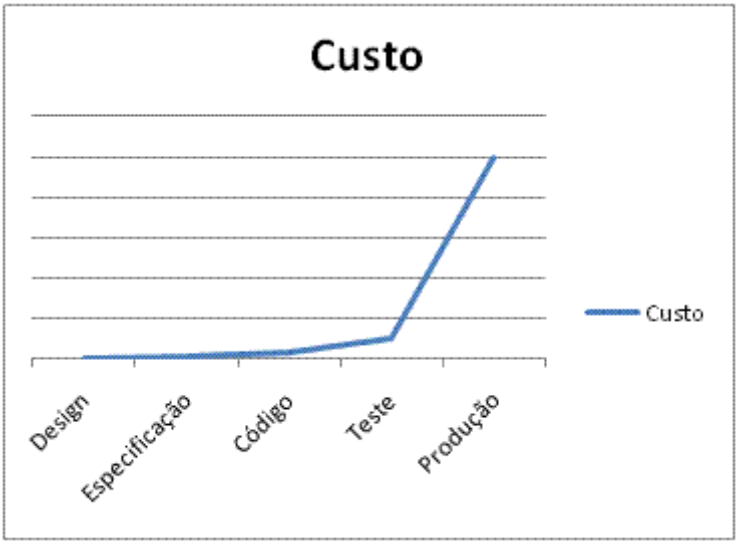


Figura 1- Regra 10 de Myers: Custo da correção dos defeitos

2.2.3 Classificação dos Testes

O Swebok (2004) classifica os testes em níveis, objetivos e técnicas de teste. Teste de software é normalmente realizada em três diferentes níveis ao longo dos processos de desenvolvimento e manutenção: unitário, integração e sistema. Os objetivos de teste podem ser projetados para verificar se as especificações funcionais estão corretas, esses são os testes funcionais, e a verificação das propriedades não funcionais como desempenho, confiabilidade, facilidade de utilização, dentre outras é conhecida como testes não funcionais. E por último, as técnicas de teste, que têm sido desenvolvidas para revelar o maior número de falhas possíveis, às vezes essas técnicas são classificadas como caixa-branca, também chamada glass-box, caixa-preta e uma última categoria que é a combinação das duas técnicas.

Na elaboração do planejamento do teste, segundo Crespo (2004), uma das etapas é a elaboração da estratégia de teste. A estratégia de teste compreende a definição dos seguintes itens: O nível de teste, isto é, a definição da fase do desenvolvimento do software em que o teste será aplicado; A técnica de teste a ser utilizada; O critério de teste a ser adotado; O tipo de teste a ser aplicado no software.

A Figura 2 ilustra graficamente a ligação existente entre os níveis de teste, as técnicas de teste, os tipos de teste e os critérios de teste que podem ser adotados ao se definir uma estratégia de teste (CRESPO 2004).

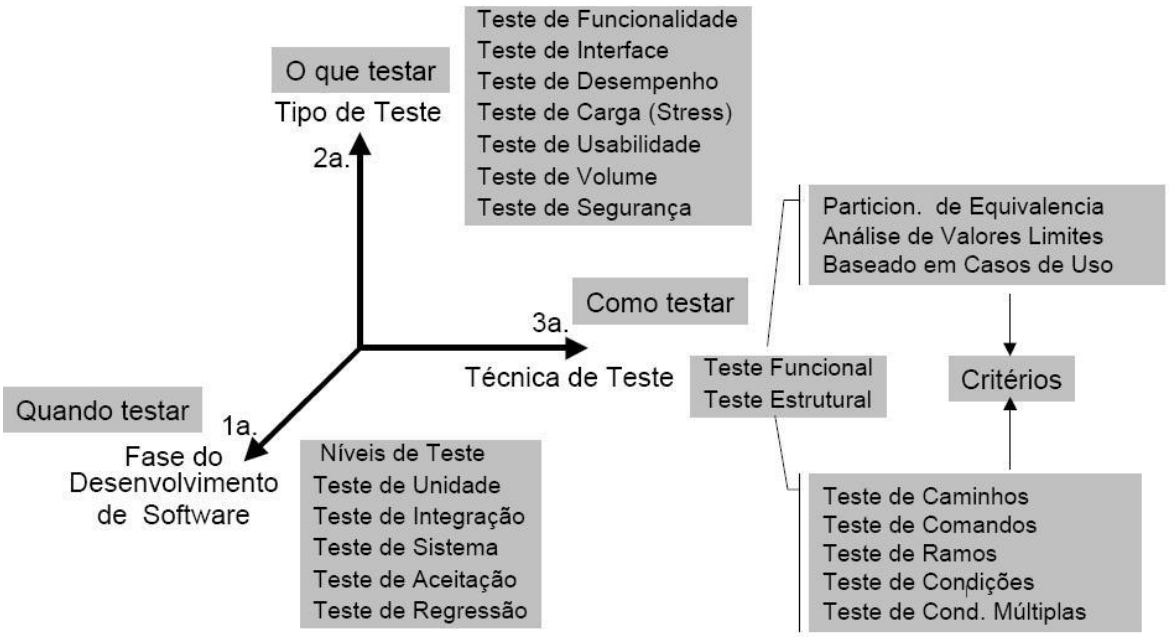


Figura 2 - Relação entre níveis, técnicas e tipos de teste (CRESPO, 2004)

2.2.3.1 Níveis de Teste de Software

De acordo com Villas Boas (2007), a atividade de teste dentro de um projeto de software pode ser classificada em quatro níveis de atuação, dependendo do objeto que está sendo testado:

(a) Teste de Unidade; (b) Teste de Integração; (c) Teste de Sistema; (d) Teste de Aceitação.

A seguir apresentam-se mais detalhes sobre esses níveis:

a) Teste de Unidade

Segundo Villas Boas (2007), o teste de unidade é o primeiro e mais básico teste que o software de um projeto pode sofrer. Deve-se lembrar que unidade, aqui, é entendida como sendo um componente indivisível de código sob a responsabilidade de apenas um programador e dividido em duas fases. A primeira se concentra em depurar a unidade em seu contexto léxico-sintático (por exemplo, por meio do uso de compiladores); nesse ponto, o uso de listas de verificação (check-list) e inspeção visual do código também traz bons resultados. Já a segunda tem por objetivo verificar a lógica da unidade. Nessa fase, a proposta é exercitar as interfaces, os comandos iterativos, os comandos condicionais, a manipulação das estruturas de dados e verificar o fluxo de controle.

Como cada unidade é testada separadamente, para Delamaro *et al*. (2007), o teste de unidade pode ser aplicado à medida que ocorre a implementação das unidades e pelo próprio desenvolvedor, sem a necessidade de dispor-se do sistema totalmente finalizado.

b) Teste de Integração

No teste de Integração segundo Delamaro *et al*. (2007), que deve ser realizado após serem testadas as unidades individualmente, a ênfase é dada na construção da estrutura do sistema. À medida que as diversas partes do software são colocadas para trabalhar juntas, é preciso verificar se a interação entre elas funciona de maneira adequada e não leva a erros.

Com relação à integração de componentes, Koscianski *et al*. (2007) relata que existem duas vertentes principais para o desenvolvimento de software: as abordagens *bottom-up* e *top-down.*

Koscianski *et al*. (2006) diz que na abordagem *bottom-up,* o programa é desenvolvido a partir de rotinas básicas que prestam serviços a rotinas de nível mais alto. Por exemplo, uma verificação de validade de CPF pode ser chamada em vários pontos de um programa. Será, então uma das primeiras a serem implementadas. E na abordagem *top-down*, faz-se o inverso; o programador trabalha supondo que o código de baixo nível esteja pronto. Assim, podem-se codificar chamadas à verificação de CPF, mesmo sabendo que ela ainda não existe. Em seu lugar, pode haver uma rotina "fantasma" (stub), que apenas retorna sempre um valor verdadeiro.

c) Teste de Sistema

De acordo com VILLAS BOAS (2007), este é o teste executado com o software completo antes que ele seja entregue ao cliente. Ele deve garantir que os requisitos do cliente (estabelecidos na Especificação de Requisitos de Software) foram implementados. Nesse teste, as interfaces externas são verificadas e demonstra-se que o software satisfaz aos requisitos funcionais e de desempenho especificados.

E DELAMARO *et al*. (2007) define que o objetivo do Teste de Sistema é verificar se as funcionalidade especificadas nos documentos de requisitos estão todas corretamente implementadas. Aspectos de correção, completude e coerência devem ser explorados, bem como requisitos não funcionais como segurança, performance e robustez. Muitas

organizações adotam a estratégia de designar uma equipe independente para realizar o teste de sistemas.

d) Testes de Aceitação

O Teste de Aceitação de acordo com Koscianski *et al*. (2007) é realizado com o propósito de avaliar a qualidade externa do produto e, na medida do possível também a qualidade em uso. Assim, só é possível quando o software está concluído e pronto para ser implantado. Evidentemente, é um teste com forte relação com o cliente, que participa do planejamento e realização dessa atividade.

Vasconcelos *et al*. (2006) classifica os testes de aceitação em dois tipos:

- Testes alfa: são feitos são feitos por um determinado cliente, geralmente nas instalações do desenvolvedor, que observa e registra os erros e/ou problemas;

- Testes beta: são realizados por possíveis clientes, em suas próprias instalações, sem a supervisão do desenvolvedor. Cada cliente relata os problemas encontrados ao desenvolvedor, posteriormente.

Alguns autores ainda consideram um quinto nível de teste que é o Teste de Regressão. Segundo Delamaro *et al* (2007), esse tipo de teste não se realiza durante o processo "normal" de desenvolvimento, mas sim durante a manutenção do software. A cada modificação efetuada no sistema, após a sua liberação, corre-se o risco de que novos defeitos sejam introduzidos. Por esse motivo, é necessário, após a manutenção, realizar testes que mostrem que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados (se for o caso) funcionam como o esperado e que os requisitos anteriormente testados continuam válidos.

2.2.3.2 Técnicas de Testes de Software

Segundo Peters & Pedrycz (2001), o teste de software envolve várias estratégias/técnicas/abordagens de teste. Elas incluem um Teste Estático versus o Teste Dinâmico (teste de caixa branca versus o teste de caixa preta), descritos a seguir:

Teste Estático

Tosetto (2004) define que teste estático, também conhecido como teste não computacional e revisão de software, não envolve a execução propriamente dita do programa. As técnicas de teste estático se valem da leitura ou inspeção visual de programas por um grupo de pessoas com o objetivo de encontrar erros. O custo de correção de erros encontrados por esse método parece ser mais baixo visto que a origem precisa do erro é localizada, enquanto que os testes dinâmicos acusam apenas sintomas dos erros. Tosetto (2004) ainda afirma que estes métodos não são eficazes na detecção de erros de modelagem ou análise de requisitos e relata que alguns autores não consideram que as atividades estáticas sejam de fato atividades de Teste de Software, mas uma revisão técnica formal.

Exemplos clássicos de técnicas de teste estático são as inspeções de código, o percorrimento e as avaliações por pares ou *peer-reviews* (MYERS, 2004).

Teste Dinâmico

O teste dinâmico também denominado análise dinâmica requer que o software seja executado com os dados do teste. Ele se baseia na utilização de provas inseridas em um programa. As provas podem ser apenas uma simples instrução de saída para rastrear valores de variáveis durante a execução do programa ou pode ainda chamar as rotinas análise que acompanham o número de vezes que os elementos de um programa são executados. (PETERS & PEDRYCZ 2001)

As técnicas dinâmicas são usadas para detectar defeitos e avaliar a qualidade do software (BURNSTEIN, 2003).

As principais técnicas de testes dinâmicos são: teste de caixa-preta e teste de caixa–branca também conhecido como teste estrutural**,** relatadas a seguir:

Teste de caixa–preta

Segundo Villas Boas (2007), o teste de caixa preta, também conhecido como funcional é realizado olhando-se o software apenas através de suas interfaces, portanto, testando sua funcionalidade. Os erros encontrados através dessa técnica pertencem às

seguintes categorias: funções incorretas, erros de interface, erros na estrutura de dados ou no acesso a dados externos, erros de desempenho e erros e erros de inicialização ou finalização.

Algumas estratégias apresentadas por Pressman (2006) deste tipo de teste são: (a) Particionamento de Equivalência, (b) Análise do Valor Limite e (c) Grafo de Causa-Efeito/Tabela de Decisão, expostas abaixo:

a) **Particionamento de Equivalência**

Particionamento de equivalência é um método de teste caixa-preta que divide o domínio de entrada de um programa em classes de dados a partir do qual os casos de teste podem ser derivados para se ter o menor número de casos possíveis. Uma classe de equivalência representa um conjunto de dados válidos ou inválidos para condições de entrada. Tipicamente, uma condição de entrada é um valor numérico específico, uma série de valores, um conjunto de valores relacionados, ou uma condição booleana, e essa condição de entrada representaria a classe toda (PRESSMAN, 2001).

b) **Análise do Valor Limite**

Com a experiência, os testadores percebem que muitos defeitos ocorrem diretamente nos limites das classes de equivalência. A análise de valor limite, que é um complemento ao critério particionamento em classes de equivalência, exige que o testador selecione elementos próximos às bordas, de modo que ambos os limites superior e inferior de uma classe de equivalência sejam cobertos pelos casos de teste (BURNSTEIN, 2003).

c) **Grafo de Causa-Efeito/Tabela de Decisão**

A grande fraqueza com o particionamento de classe de equivalência é que ela não permite que os testadores explorem combinações das condições de entrada. As combinações podem ser cobertas por alguns casos de casos de teste gerados a partir das classes. Grafo de Causa e efeito é uma técnica que pode ser usada para combinar as condições e obter um conjunto eficaz de casos de teste que pode revelar inconsistências na

especificação. No entanto, a especificação deve ser transformada em um gráfico que se assemelha a um circuito de lógica digital. O gráfico deve ser convertido em uma tabela de decisão que o testador utiliza para desenvolver casos de teste (BURNSTEIN, 2003).

Teste estrutural ou teste de caixa–branca

Teste estrutural ou teste de caixa-branca é a técnica que permite examinar a estrutura interna do programa, ou seja, a lógica do programa e freqüentemente, erros de especificação (MYERS, 2004).

Pressman (2006) relata que usando métodos de teste caixa-branca, o engenheiro de software pode derivar casos de teste que garantam que: todos os caminhos independentes de um módulo tenham sido exercitados pelo menos uma vez, exercitem todas as decisões lógicas em seus lados verdadeiro e falso, executem todos os ciclos nos seus limites e dentro de seus intervalos operacionais e exercitem as estruturas de dados internas para garantir sua validade.

2.2.3.3 Tipos/Categorias de Testes de Software

De acordo com Peters & Witold (2001), levando-se em consideração a diversidade do teste de software existente, é vantajoso considerar os tipos de testes, à medida que se tornam disponíveis a um projetista. Isso também ajudará a identificar a abrangência de um determinado teste e a explicar as principais vantagens e desvantagens, além de esclarecer o desenvolvedor sobre as suas limitações.

Os tipos de teste conhecidos são os Testes Funcionais e Testes Não Funcionais, apresentados abaixo:

Testes Funcionais:

De acordo com Myers (2004) os testes funcionais são um processo de tentativa para encontrar discrepâncias entre o programa e as especificações externas que são descrições precisas do funcionamento do programa do ponto de vista do usuário final. Os testes funcionais usados em pequenos programas são uma atividade de caixa preta.

Não Funcionais:

A fase dos testes não funcionais é especialmente utilizada para detectar defeitos externos de interface de hardware e software (BURNSTEIN, 2003).

Segundo Magela (2006) os testes funcionais não fornecem a garantia necessária para um software entrar em produção e exemplifica da seguinte forma: embora todos os requisitos do usuário possam ser atendidos, se para cada tarefa o usuário esperar dez minutos para sua execução, o software está com problema de desempenho e sem dúvida, esse problema inviabiliza a liberação do software, por isso os testes não-funcionais são necessários.

São apresentados a seguir alguns exemplos de testes não-funcionais: usabilidade, volume, ambiente, segurança, desempenho, instalação e recuperação.

Usabilidade

A usabilidade segundo Ferreira (2002) está relacionada à eficácia e eficiência da interface diante do usuário e pela reação do usuário diante da interface.

Myers (2004) lista alguns motivos para a usabilidade ser testada:

- Cada interface do usuário foi adaptada ao entendimento, fundo e ambiente claros e objetivos para o usuário final?

- São as saídas de programa de fácil entendimento, não agressoras, e com uma linguagem simples, não de computador?

- As mensagens de erro são diretas e de fácil compreensão?

- O conjunto total de interfaces de usuário apresenta considerável integridade conceitual, uma coerência subjacente, e uniformidade da sintaxe, convenções, semântica, formato, estilo e abreviaturas?

- Onde a precisão é fundamental, como em um sistema bancário on-line, está presente excesso de dados suficiente na entrada?

- O sistema contém um número excessivo de opções, ou opções que são improváveis de ser utilizados? Uma tendência nos software modernos é apresentar ao usuário somente as opções de menu que têm maior probabilidade de usar, baseada em testes de software e considerações do *design*;

- O programa é fácil de usar? Por exemplo, é o caso de uma entrada diferente sem que esteja claro para o usuário? Além disso, se um programa requer navegação através de uma série de menus ou opções, é claro como retornar para o menu principal? Pode um usuário facilmente subir ou descer um nível?

Volume

No Teste de Volume, de acordo com Myers (2004), o programa é submetido a pesados volumes de dados.

Magela (2006) completa dizendo que este tipo testa a capacidade do sistema de suportar um grande volume de dados sem preocupação nenhuma com o tempo, que é o teste de carga.

Ambiente

Os testes de ambiente suportam os requisitos de ambiente. Limitações como temperatura local, umidade e interferência dentre outros pontos devem ser verificadas e validadas (MAGELA, 2006).

Segurança

Os testes de segurança garantem a confidencialidade das informações e a proteção dos dados contra o acesso indevido de terceiros. A quantidade de segurança fornecida depende dos riscos associados. A garantia de confidencialidade das informações é desenhada para proteger os recursos da organização. Algumas informações, se reveladas ou adulteradas, podem comprometer o negócio ou aumentar a vantagem competitiva dos concorrentes (BASTOS *et al*., 2007)

Atualmente, com a Internet, seguranças extras devem ser fornecidas. Contudo, questões como firewall, Proxy e outros pontos não dizem respeito ao software e sim à empresa. Portanto, é uma responsabilidade da empresa atuar na chamada zona militarizada, e não do software em desenvolvimento (MAGELA, 2006).

Desempenho (performance)

O teste de Performance é projetado para testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado. O teste de performance ocorre ao longo de todas as etapas do processo de teste. Mesmo ao nível da unidade, o desempenho de um módulo individual pode ser avaliado como testes caixa-branca são conduzidas (SWEBOK, 2004).

Instalação

Alguns tipos de sistemas de software têm os procedimentos de instalação complicados. Testar o procedimento de instalação é uma parte importante do processo de teste do sistema. Isto é particularmente verdade em um sistema com instalação automatizada que é parte do pacote do programa. Um mau funcionamento da instalação do programa poderia impedir o usuário de ter uma experiência de sucesso com o sistema principal que são cobrados com os testes. A primeira experiência do usuário é quando ele instala o aplicativo. Se nesta fase algum erro acontece, então o usuário / cliente pode achar que o produto não é confiável (MYERS, 2004)

Recuperação

Avalia o tempo de recuperação do software quando seus recursos falham. Por exemplo, um ambiente com mais de um servidor de aplicação poderá ter um dos servidores desligado. Iremos monitorar se o outro servidor de aplicação assume corretamente a função desse último; e o mesmo se aplica a banco de dados e a outros recursos necessários ao correto funcionamento do software. (MAGELA, 2006).

2.2.4 Processo de Teste de Software e Documentação

Bastos *et al*. (2007) diz que a melhor maneira de testar um software é ter um processo de teste claramente definido, e o objetivo de um processo de teste, com metodologia própria, é minimizar os riscos causados por defeitos provenientes do processo de desenvolvimento.

O processo de teste apóia as atividades de teste e fornece orientações para as equipes de teste, desde o planejamento de teste até a avaliação final dos testes, de modo a proporcionar garantias que justifiquem que os objetivos dos testes serão cumpridos de forma eficaz (SWEBOK, 2004).

Segundo Villas Boas (2007), o Processo de Teste é caracterizado por um conjunto de atividades que são executadas ao longo de todo o ciclo de desenvolvimento do software. Villas Boas (2007) agrupa essas atividades em etapas bem definidas, que são:

1. Etapa de planejamento e controle

Consiste em elaborar a Estratégia de Teste e o Plano de Teste a ser utilizados de modo a minimizar os principais riscos do negócio e fornecer os caminhos para as próximas etapas (BASTOS *et al*., 2007).

2. Etapa de Projeto

Projetar é especificar de maneira detalhada a forma de se realizar algo. Nessa etapa será escolhido um grupo específico de características a serem testadas, descrevendo detalhadamente os métodos que serão empregados e os testes que deverão ser feitos. O resultado final dessa etapa é o documento Projeto de Teste (PJ) (Villas Boas, 2007).

3. Etapa de Implementação

Implementar é fornecer os subsídios indispensáveis à execução de alguma tarefa. Na etapa de implementação é que se descrevem detalhadamente os itens que serão testados, as entradas utilizadas, as saídas esperadas e os passos necessários para se executar os testes, ou seja, a seqüência de tarefas necessárias para se analisar um item de software com a finalidade de se avaliar um conjunto de suas características. O resultado final dessa etapa será o documento Caso-de-Teste (CS) e Procedimento de Teste (PR) (Villas Boas, 2007).

4. Etapa de Execução

Essa etapa é a parte prática do teste, isto é, aquela em que se efetiva a tarefa de teste propriamente dita. Aqui, a partir do procedimento de teste que será seguido e dos casos-de-teste associados, o testador realizará os testes. Nessa etapa, ele deve registrar toda a atividade de teste, identificando data e hora do teste, autor do teste, o procedimento

seguido, o pessoal envolvido, os resultados obtidos, as condições ambientais e os eventos não esperados (quando ocorrerem). Caso aconteçam problemas durante os testes, estes deverão ser relatados, sendo referidos o procedimento em uso, os casos-de-teste associados e os itens que estão sendo testados, fornecendo uma descrição dos problemas, as entradas utilizadas, data e hora do ocorrido, o passo do procedimento, os observadores e impacto do problema sobre os testes. Ao final das atividades de teste, um resumo deve ser fornecido referindo os itens testados, sumarizando os resultados obtidos e fornecendo uma avaliação baseada nesses resultados. Os produtos gerados nessa etapa são os documentos Registro de Teste (RG), Relatório de Problemas de Teste (RT) e Relatório de Resumo de Teste (RE) (Villas Boas, 2007).

E Villas Boas (2007) complementa dizendo que embora essas etapas ocorram em sequência, nem sempre elas acontecem em bloco para cada teste; na verdade, elas são diluídas ao longo do ciclo-de-vida do projeto.

Na figura 3 Molinari (2008), exemplifica uma das diversas formas de representar um processo de teste com uma visão geral de um processo.

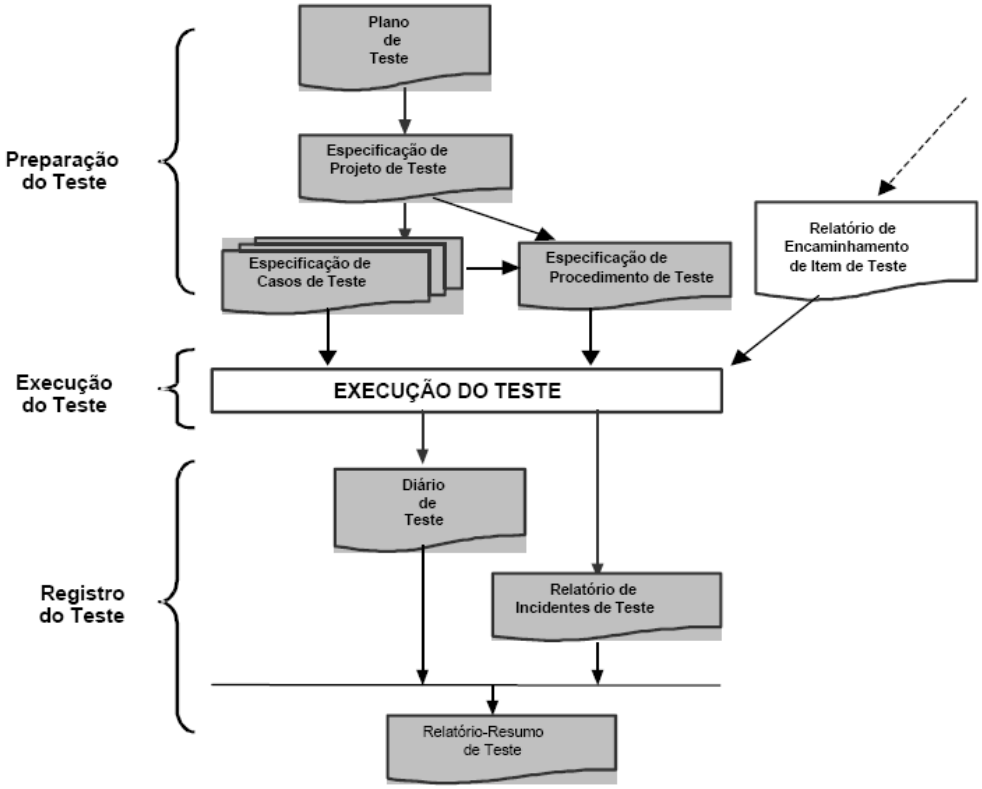


Figura 3 - Exemplo de processo de teste: visão geral (MOLINARI, 2008)

2.2.4.1 Documentação

Na etapa de teste, durante o ciclo de vida do projeto, são gerados uma série de documentos que servem para acompanhamento e controle do projeto e existem vários modelos de documentos de teste sendo hoje utilizados (Villas Boas,2003).

A seguir alguns dos principais documentos utilizados no processo de teste segundo a IEEE 829-1998:

- Plano de Teste: determina o escopo, abordagem, recursos e cronograma das atividades de teste. É onde se identifica os itens a serem testados, as características a serem testadas, as tarefas de testes a serem realizadas, os responsáveis por cada tarefa, e os riscos associados com o plano.
- Projeto de Teste: refina a abordagem de teste e identifica os recursos a serem cobertos pelo projeto e associados aos testes. Ele também identifica os casos de teste e procedimentos de teste.
- Documento de Caso de Teste: especificam valores reais utilizados para a entrada juntamente com os resultados esperados. Um caso de teste também identifica as restrições sobre os procedimentos de teste resultante da utilização do caso de teste específico.
- Procedimento de Teste: os procedimentos de teste identificam todos os passos necessários para operar o sistema e usar os casos de teste especificados, a fim de implementar o projeto de teste associado.

E ainda segundo a IEEE 829, os relatórios e registros dos testes são cobertos por quatro documentos:

- Um relatório de transmissão de item de teste identifica os itens de teste que estão sendo transmitidos para o teste no evento que separa o desenvolvimento e grupos de teste estão envolvidos, ou no evento em que o início formal da execução de um teste seja desejado
- Um registro de teste é utilizado pela equipe de teste para gravar o que ocorreu durante a execução do teste.
- Um relatório do incidente de teste descreve qualquer evento que ocorre durante a execução do teste, que exige mais investigação.
- Um resumo do relatório de teste resume as atividades de teste associados

com uma ou mais especificações do projeto de teste.

A figura 4 mostra a relação de documentos de teste para o processo de teste segundo a IEEE 829

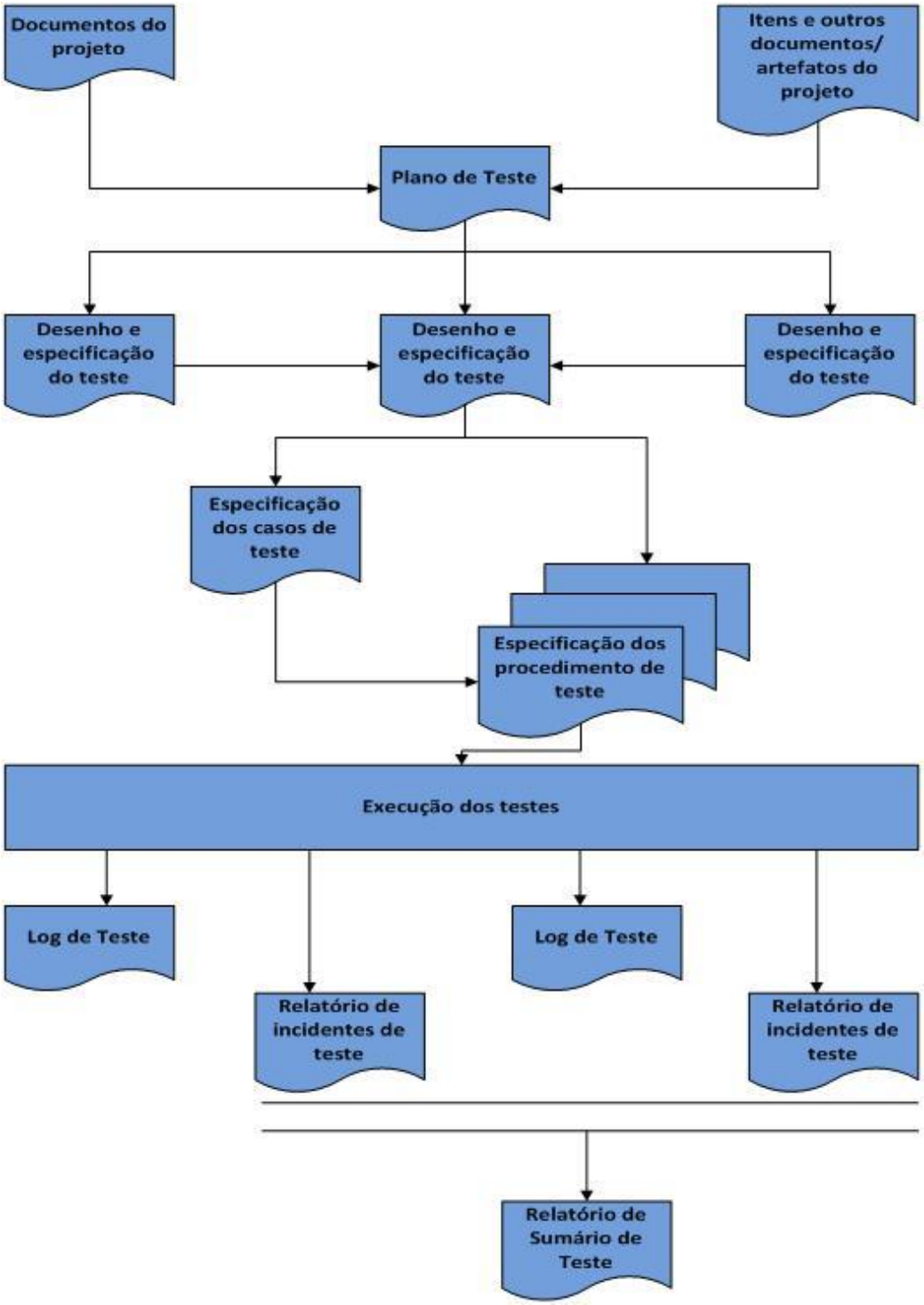


Figura 4 - Relação de documentos de teste para o processo de teste - IEEE 829

2.2.5 Ferramentas para Testes de Software

Ferramentas de teste são os suportes utilizados por indivíduos responsáveis por testes para cumprir essa responsabilidade. As ferramentas abrangem um vasto leque de atividades e são aplicáveis para uso em todas as fases do ciclo de vida de desenvolvimento de sistemas. Algumas das técnicas são manuais, algumas automatizadas; algumas realizam testes estáticos, outras dinâmicos, alguns avaliam a estrutura do sistema e outras, a função do sistema. (Perry, 2006)

As ferramentas são importantes para reduzir a intervenção humana na atividade de teste, aumentando a produtividade e influenciando na confiabilidade do software testado. Elas também fornecem suporte para o teste de regressão, já que os casos de teste utilizados numa dada sessão de teste podem ser, facilmente, obtidos para a revalidação de um software após uma alteração (Villas Boas, 2003).

A habilidade necessária para usar as ferramentas assim como os seus custos de execução variam significativamente. Algumas das habilidades são altamente técnicas e envolvem um conhecimento profundo em programação e o sistema que está sendo testado. Outras ferramentas são de natureza geral e são úteis para quem testa, mas não é o responsável direto por ele. Algumas técnicas envolvem apenas os custos homem-hora, enquanto outros devem ser realizadas por uma equipe e fazem uso pesado de recursos computacionais no processo de teste. (Perry, 2006)

Conforme Almeida (2007), para comprar ou desenvolver uma ferramenta para automatizar um teste, deve-se antes fazer algumas perguntas básicas como:

- Qual é o custo/benefício da ferramenta?
- O que ela vai agregar no ciclo de vida dos projetos?
- Qual será o número de colaboradores que a utilizarão?
- Existe uma metodologia que é seguida e onde essa ferramenta poderá ser encaixada?
- No processo de desenvolvimento, poderá existir uma curva de aprendizado para utilização da ferramenta?
- Ela reduzirá de alguma forma o tempo e custo no processo de desenvolvimento? Haverá um retorno do investimento?
- Foi feito um estudo para se constatar que a ferramenta atenderá as expectativas da empresa, bem como se suas funcionalidades e atribuições se

enquadram dentro da metodologia e processo adotado?

Se todas as perguntas tiveram como resposta “Sim”, então talvez a ferramenta agregue valor e infelizmente, grande parte das empresas erra nesse ponto, pois adquirem uma série de ferramentas antes mesmo de saber onde elas se encaixarão no processo de desenvolvimento e testes e até mesmo, e sem saber ao certo quais serão os benefícios que elas poderão trazer. (Almeida, 2007).

Algumas ferramentas conhecidas são: Mantis, *Testlink* e *Selenium*, brevemente descritas a seguir:

Mantis

O Mantis é uma ferramenta *Open Source* cujo principal objetivo é registrar e acompanhar os *bugs* encontrados em um projeto, desde o seu nascimento até o seu fechamento. Neste cenário, o ciclo de vida gerenciado pelo Mantis inicia-se quando um *bug* é registrado; as fases seguintes são empregadas para a confirmação, correção, revisão e o fechamento do *bug*. Cada *bug* recebe um número sequencial único a fim de identificá-lo durante as operações de consulta, relatórios e modificações. Por outro lado, diversas informações são agregadas ao *bug* durante as fases que compõem o seu ciclo de vida, como por exemplo, status, prioridade, severidade, comentários, anexos, etc (http://www.mantisbt.org/).

Testlink

O *Testlink* é uma ferramenta Open Source que pode ser integrada com outras ferramentas de gerenciamento de *bugs* como o Mantis. O *TestLink* permite que o usuário crie os planos e casos de teste. Os planos de teste podem ser atribuídos a um determinado integrante da equipe, para que execute ou acompanhe o resultado dos testes. Permite também a geração de relatórios gerenciais e estatísticos sobre os testes executados (http://blog.testlink.org/).

Selenium

O *Selenium* é uma ferramenta Open Source usada para a criação de testes de

regressão automatizados para aplicações WEB. Os testes rodam diretamente a partir do navegador. Na realidade, em virtude desta característica do *Selenium*, os testes podem rodar virtualmente em qualquer navegador que suporte Java Script como Internet Explorer, Firefox, Opera, Safari, Konqueror, etc (http://seleniumhq.org/)

2.3 MODELOS DE MATURIDADE EM TESTE DE SOFTWARE

A seguinte secção apresenta alguns modelos de qualidade do processo de teste e apresenta detalhadamente o nível 1 do MPT.BR.

Segundo Rouiller *et al*. (2004), diversos modelos e normas de qualidade de software vêm sendo propostos ao longo dos últimos anos. Esses modelos e normas têm sido fortemente adotados por organizações em todo mundo. Entretanto Corso (2008) relata que embora os modelos de maturidade de software existentes tenham grande aceitação e aplicação pela indústria de software, eles não abordam adequadamente as questões relacionadas aos testes e nem possuem um processo de teste bem definido, por isso, vem sendo estudado modelos relacionados à área de teste.

Molinari (2003), afirma que esta nova abordagem em garantir qualidade de software alinhada ao processo de testes e modelos, faz com que o teste de software deixe de ser mais uma atividade no processo de desenvolvimento para ter suas próprias metodologias. Assim, permite que sua estrutura varie de acordo com as características e necessidades da empresa.

Com isso, pesquisadores iniciaram um processo de desenvolvimento de modelos de maturidade específicos para a atividade de teste que completassem as deficiências nesta área. Alguns dos modelos de maturidade de teste desenvolvidos são: TIM - *Test Improvement Model*, definido por Ericson, Subotic e Ursing , Software Testing Maturity Model (SW-TMM ou TMM) criado pelo *Illinois Institute of Technology* em 1996, Test Process Improvement (TPI) , desenvolvido por Koomen e Pol em 1997, entre outros modelos já disponíveis. (Santos, 2006).

Uma maneira de adequar o processo de teste de software às boas práticas do teste é adotar um modelo de melhoria que atenda a área de teste de software. (REFFSON *et al*. 2006)

2.3.1 TPI - TEST PROCESS IMPROVEMENT

O Test Process Improvement (TPI) foi desenvolvido por Koomen e Pol, em 1997. Segundo Santos (2006) é o modelo mais utilizado na Europa, e a principal razão para o seu desenvolvimento foi o fato de o processo de teste ser considerado importante, porém de difícil controle. O modelo foi desenvolvido para tornar definição e melhoria de processos

de teste mais fáceis e controladas.

Diferentemente do TMM ele organiza o conhecimento em áreas chave e não em níveis de maturidade. Existem 20 áreas chave no TPI.

Dentro de cada área existem níveis diferentes indicando maior maturidade naquela área. Dentro de cada nível temos checkpoints (pontos de verificação) e sugestões de melhorias. Os checkpoints servem para indicar os pontos fortes e os pontos fracos de cada área. As sugestões de melhorias orientam a passar de um nível para outro.

2.3.1.1 – Áreas chave do Test Model Improvement

Cada área chave pode ser definida da seguinte forma, conforme Sogeti (2007).

A definição de cada área chave do TPI é realizada contemplando: (1) estratégia de testes; (2) modelo do ciclo de vida; (3) momento de envolvimento; (4) Estimativa e planejamento; (5) Técnicas de especificação de testes; (6) Técnicas de testes estáticos; (7) Métricas; (8) Ferramentas/Automação de teste; (9) Ambiente de teste; (10) Ambiente de escritório; (11) Comprometimento e comunicação; (12) Funções de testes e treinamento; (13) Escopo de metodologia; (14) Comunicação; (15) Relatórios/Reporte; (16) Gerência de desvios/falhas; (17) Gerência de *testware* (artefatos do teste); (18) Gerência do processo de testes; (19) Avaliação; (20) Testes de baixo nível, apresentadas a seguir, conforme Sogeti (2007). Cada uma delas é descrita a seguir (SOGETI, 2007):

A **estratégia de testes** define que deve-se focar na busca de falhas mais importantes de forma mais rápida e barata. Deve definir quais requerimentos e quais riscos serão cobertos pelos testes.

Já o **modelo de ciclo de vida** define as fases como, planejamento, preparação, especificação, execução e finalização. Em cada fase várias atividades são executadas. Para cada atividade, os seguintes aspectos devem ser definidos: o propósito, os inputs, o processo, os outputs, as dependências, as técnicas e ferramentas aplicáveis, as facilidades requeridas, a documentação e etc.

O **momento de envolvimento** diz que o momento de iniciar o teste deve ser o mais cedo possível, ou seja, durante o desenvolvimento ou até durante o início do projeto, e não após o sistema estar pronto. Iniciar os testes junto com o desenvolvimento ajuda a evitar falhas ou encontrá-las mais rápida e facilmente e também propicia ajustar os diferentes

testes a serem feitos.

A **estimativa e planejamento** definem quais atividades têm que ser executadas e que recursos (humanos) são necessários para cada uma. Bons planejamentos e estimativa são muito importantes, pois são a base para alocação de recursos. São importantes também para cumprimento de prazos.

A definição de **Técnicas de Especificação de Testes** é "uma forma padronizada de descobrir test cases através de informações (entrevistas, documentações, etc.)". A aplicação destas técnicas no leva a "insights" sobre a qualidade e profundidade dos testes, além de tornar o teste reutilizável (padronização).

A área chave **Técnicas de testes estáticos** define que nem todo programa pode ser testado automaticamente. Inspecionar um produto sem rodar nenhum programa ou avaliá-lo de acordo com um certo grau de qualidade é chamado de Teste Estático (Testes Manuais). Checklists (requisitos) são muito úteis para isso.

Métricas são observações quantitativas sobre características de um produto ou processo. Para o processo de teste, métricas do progresso do processo e do progresso da qualidade do produto são muito importantes. São utilizadas para controlar o processo de teste, para substanciar os avisos de teste e também para possibilitar comparar sistemas e processos.

O uso de **ferramentas/automação de teste** em processos de teste existe de várias formas e tem, em geral, um ou mais dos seguintes objetivos: diminuir tempo de teste, aumentar a flexibilidade do teste, testes mais profundos, maior quantidade de combinações, maior motivação dos testadores, etc..

Existe **ambiente de teste** de acordo com a demanda. Este ambiente é composto principalmente pelos seguintes componentes: hardware, softwares de apoio, formas de comunicação, facilidade de montar ou utilizar banco de dados, etc. O ambiente deve ser composto e disponibilizado de forma a otimizar o resultado dos testes e a atender as necessidades do objeto a ser testado.

A área chave **ambiente de escritório** define que o local de trabalho dos testadores também é muito importante (salas, móveis, Computadores, impressoras, telefones, etc.). Um ambiente adequado, agradável e disponibilizado a tempo motiva as pessoas e facilita a comunicação interna e externa, tornando o trabalho mais eficiente.

Comprometimento e motivação das pessoas envolvidas é muito importante para uma boa fluência no processo de teste. A todas elas, gerente ou testador deve ser atribuída

muita importância, pelo menos para que seja viabilizada criatividade sobre boas condições de teste. Elas devem contar com tempo, dinheiro e recursos para otimização de seus resultados.

As **funções de testes de treinamento** definem que num processo de teste testes a composição correta do "time" de testadores é muito importante. Diferentes formações, conhecimentos e habilidades são fundamentais. Além de expertise em testes, conhecimento sobre o negócio, a instituição e conhecimentos gerais sobre TI também são necessários. Habilidade no trato com pessoas (cliente e funcionários) também é muito importante. Para isso ser obtido, treinamento é fundamental.

Um **escopo de metodologia** deve ser utilizado para cada processo de teste na organização, incluindo atividades, procedimentos, regulamentos, técnicas e etc.. Metodologias muito diferentes ou muito genéricas têm efeitos negativos neste processo. O objetivo é que a organização utilize metodologia suficientemente genérica para que possa ser aplicada em qualquer situação, mas esta metodologia deve conter detalhes suficientes para que não seja necessário repensar processos a toda hora.

Num processo de teste, **comunicação** com todas as pessoas envolvidas é muito importante. Tanto com o time de testadores, como com os desenvolvedores, usuários, clientes e etc. Estas formas de comunicação são importantes para facilitar o processo teste, para criar boas condições e otimizar a estratégia de teste. Assim como também deixar todos informados sobre o progresso e qualidade do produto.

Reportar/Relatar não é somente apresentar a lista de falhas detectadas, mostrando o nível de qualidade do produto. O reporte deve também informar, recomendar e aconselhar os desenvolvedores e o cliente sobre o produto em foco.

A **gerência de desvios/falhas** relata que apesar do gerenciamento das falhas ser de responsabilidade do projeto, e não especificamente dos testadores, este está muito ligado a Equipe de Testes. Um bom gerenciamento deve ser capaz de mostrar o ciclo de vida das falhas e sugerir melhoramentos substanciados pela falhas detectadas. Uma boa Equipe de Testes deve acompanhar e cobrar o tempo de conserto das falhas (pelo menos mostrar estatisticamente).

A **gerência de testware (artefatos do teste)** define que os produtos Os produtos dos testes devem ser passíveis de manutenção, de serem reutilizados e de serem gerenciados. Além de produtos como plano de testes, especificações, base de dados, e arquivos, é importante também que seja gerenciado o processo anterior, como a definição

das funções. Isto é necessário, para que, no caso de uma versão errada, o processo possa ser retomado sem perdas do que já foi executado e para acompanhar a evolução do produto.

A **gerência do processo de testes** propõe que para gerenciar cada atividade e cada processo, os quatro passos são essenciais: planejar, fazer, checar e agir (influir). O gerenciamento tem importância vital na otimização do teste que é sempre realizado num processo turbulento.

Realizar a **avaliação** significa verificar os produtos intermediários do desenvolvimento, como os requerimentos ou as definições das funções. A importância da avaliação está em achar as falhas o mais cedo possível durante o desenvolvimento. Isto faz o retrabalho custar muito menos. Avaliação também pode ser fácil por não necessitar rodar programas ou disponibilizar ambiente, etc..

Os **testes de baixo nível** são aqueles realizados pelos desenvolvedores. Os mais conhecidos são o teste unitário e o testes de integração. Assim como a avaliação, estes testes detectam falhas mais cedo, ou seja, durante o desenvolvimento. São eficientes por necessitarem de pouca comunicação já que são achados e corrigidos pelos que estão produzindo o software.

2.3.1.2 Test Maturity Matrix

Como dito anteriormente, dentro de cada área existem níveis diferentes indicando maior maturidade naquela área. Segundo Conti (2007), dentro dessa concepção de áreas-chaves o modelo de testes oferece níveis (do ‘A’ para o ‘B’ para o ‘C’ para o ‘D’), estes níveis são baseados no tempo, custo e qualidade.

O TPI oferece uma ferramenta extra chamada *Test Maturity Matrix* ou Matriz de Maturidade. De acordo com Conti (2007), a partir dessa matriz, todos os níveis de maturidade e áreas são relacionados entre si. Seu propósito é mostrar o estado do processo interno de teste e definir as dependências entre os níveis de maturidade e as áreas-chave. A matriz é composta com resultados por 13 níveis de maturidade dos testes. Seu propósito é identificar os pontos fortes e fracos de um processo, permitindo a elaboração de planos de ação para melhoria.

2.3.2 TIM - TEST IMPROVEMENT MODEL

O Test *Improvement Model* (TIM) foi definido por Ericson, Subotic e Ursing (2003) e está baseado no CMM e em um outro modelo de testes TSM (*Testability Support Model*).

Ericson, Subotic e Ursing (2003) apud Santos (2006), declara que O TIM pode ser utilizado para dois propósitos: identificar o estado atual de Práticas de Áreas de Processo em um processo de uma organização e sugerir formas de fortalecer os pontos fortes de um processo e remover suas fraquezas e o divide em dois componentes: Modelo de Maturidade e Procedimento de Avaliação abaixo descritos:

2.3.2.1 Modelo de Maturidade

Segundo Santos (2006), o modelo está estruturado em Níveis de Maturidade e Áreas de Processo. Ao todo são cinco Níveis de Maturidade, o nível inicial, Nível 0, é um nível caótico e não conforme. Os níveis de 1 a 4 possuem objetivos associados, que quando atendidos proporcionam a melhoria gradual do processo de teste. Cada objetivo é composto por sub-objetivos e para que um objetivo seja atingido, todos os sub-objetivos dele precisam ser atendidos.

O TIM atualmente possui cinco áreas de processo distribuídas pelos cinco níveis. As áreas de processo representam melhorias para uma área específica no domínio de teste e se diferenciam das áreas do CMM por não estarem confinadas a um só nível.

Ainda segundo Santos (2006), os níveis de maturidade do TIM são:

- Nível 0 – *Initial*

Este é um nível inicial. Nele, o processo de testes é caótico e sem definição. Não há nenhum requisito para classificação de um processo neste nível.

- Nível 1 – *Baselining*

Neste nível são documentados padrões, políticas, procedimentos, metodologias e métodos de teste, constituindo um processo básico de testes.

- Nível 2 - *Cost-effectiveness*

Este nível busca a redução dos custos do processo de testes, com atividades relativas à treinamentos da equipe, redução de re-trabalho e reuso de artefatos existentes ou partes deles.

- Nível 3 - *Risk-lowering*

  Este nível está focado na redução dos riscos, com atividades relativas ao planejamento e gerenciamento de testes, detecção de erros perigosos e uma abordagem de planejamento e distribuição de recursos dirigida a riscos.

- Nível 4 – *Optimizing*

  Este nível se caracteriza por possuir um processo otimizado de testes, no qual são documentados e gerenciados os aspectos de teste relativos ao atendimento política de qualidade da organização. Há uma mudança cultural e os testes passam a ser vistos como uma forma de prevenir de defeitos através da busca de suas causas e definição de estratégias para eliminá-las. É incentivada e buscada a melhoria contínua do processo e os testes estão totalmente integrados ao ciclo de vida do projeto.

2.3.2.2 Procedimento de Avaliação

De acordo com Santos (2006), o procedimento de avaliação do TIM é constituído de três passos essenciais:

- O modelo e o procedimento de avaliação são explicados para uma equipe de membros da organização (é importante que membros representativos de todos os setores da organização participem);
- A avaliação é executada por meio de entrevistas com os membros selecionados. As entrevistas são guiadas por um questionário de apoio e não podem ser respondidas com sim/não, quando isso acontecer o entrevistador deve iniciar uma discussão sobre a questão;
- Os resultados das entrevistas são analisados e é elaborado um perfil de maturidade, onde o grau de satisfação do processo da organização em relação a cada área de processo é identificado.

2.3.3 TMMI – TEST MATURITY MODEL INTEGRATED

O TMMI – Test Maturity Model Integrated desenvolvido pela TMMI Foundation, segundo Veenendaal (2009) é um modelo detalhado para a melhoria do processo de teste e

é posicionado como sendo complementar ao CMMI. O desenvolvimento do TMMI utiliza o modelo TMM desenvolvido pelo *Illinois Institute of Technology* como uma das suas principais fontes e apóia as organizações a avaliar e melhorar seu processo de teste.

Níveis de Maturidade

Existem cinco níveis no TMMi que prescrevem uma hierarquia de maturidade e um caminho evolutivo para testar a melhoria do processo: Inicial, Gerenciando, Definido, Gestão e Medição e Otimizado. Cada nível possui um conjunto de áreas de processo que uma organização deve se concentrar para alcançar a maturidade nesse nível, descritos na Figura 5 a seguir:

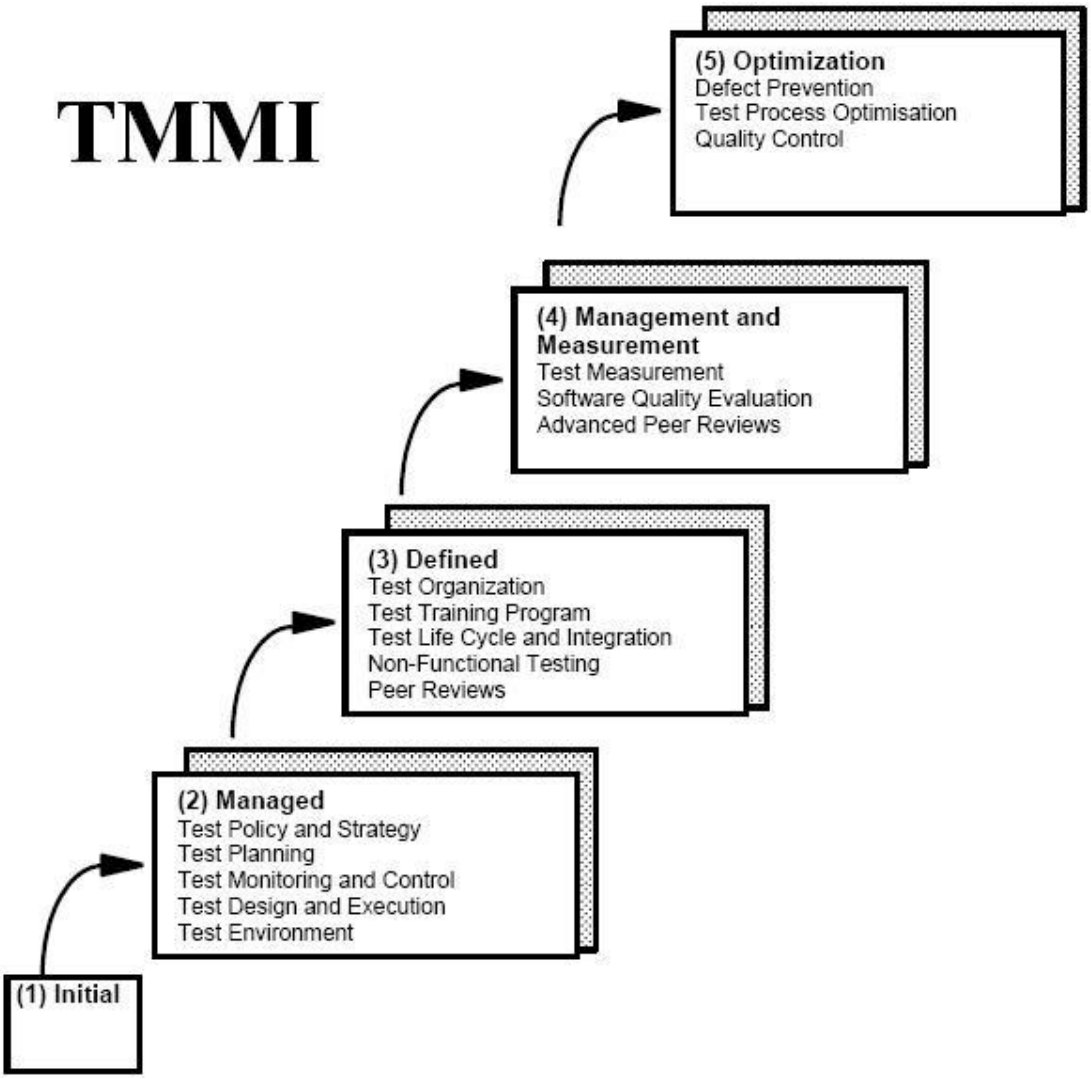


Figura 5 - Níveis do TMMI (VEENENDAAL 2009)

A seguir, a descrição dos 5 níveis de maturidade:

Inicial

No TMMI nível 1, o teste é um processo caótico, indefinido e muitas vezes é considerada como uma parte de depuração. A organização geralmente não fornece um ambiente estável para apoiar os processos de teste.

Os testes são desenvolvidos, após a codificação ser concluída.

O objetivo nesse nível é mostrar que o software é executado sem grandes falhas. Os produtos são liberados sem visibilidade adequada sobre a qualidade e riscos.

Gerenciado

No TMMI nível 2, teste se torna-se um processo gerenciado e é claramente separado da depuração. No entanto o teste ainda é visto como sendo uma fase do projeto que segue a codificação, acarretando em atrasos no início das atividades.

São definidos nessa fase, plano de teste, que é o documento que define o que será necessário nos testes, quando, como e por quem. Também são aplicadas técnicas de teste para derivar casos de teste.

As áreas de processo do nível 2 do TMMI são:

Política e Estratégia de Teste

Planejamento de Teste

Acompanhamento e Controle de Teste

Projeto e Execução de Teste

Ambiente de Teste

Definido

No TMMI nível 3 o teste é plenamente integrado no ciclo de desenvolvimento, não sendo mais uma fase que segue a codificação.

O planejamento é feito numa fase precoce do projeto, por exemplo, durante a fase de requisitos, por meio de um plano de teste principal, que se baseia no teste de habilidades de planejamento e compromissos adquiridos no TMMI nível 2.

As áreas de processo do nível 3 do TMMI são:

Organização de Teste

Programa de Treinamento de Teste

Ciclo de Integração e Testes

Testes não-funcionais

Revisões em Pares

Gestão e Medição

No TMMI nível 4 as organizações de teste são completamente definidas, bem fundamentadas e com processo mensurável. Nesse nível, a organização e projetos estabelecem objetivos quantitativos para a qualidade do produto e desempenho do processo e usa esses critérios para gerenciar. Qualidade do produto e desempenho do processo é entendido em termos estatísticos e é gerenciado ao longo do ciclo de vida.

Os produtos são avaliados/medidos através de critérios quantitativos para a qualidade de tais atributos como confiabilidade, usabilidade e manutenibilidade.

As áreas de processo do nível 4 do TMMI são:

Medição de Teste

Avaliação da Qualidade do Produto

Revisões Avançadas

Otimizado

Com base em todos os resultados que foram alcançados pelo cumprimento de todas as metas de melhoria da maturidade dos níveis anteriores, o teste é agora um processo completamente definido e capaz de controlar os custos e a eficácia dos testes.

As áreas de processo do nível 5 do TMMI são:

Prevenção de Defeitos

Otimização de Processo de Teste

Controle de Qualidade

2.3.4 MPT.BR – MELHORIA DO PROCESSO DE TESTE BRASILEIRO

O MPT.BR – Melhoria do Processo de Teste Brasileiro, segundo ALATS (2010) é um modelo que está sendo desenvolvido com o princípio básico de ser compatível, mas não necessariamente igual, com o modelo MPS.BR criado pela Sociedade Brasileira para Promoção da Exportaçao de SofTware (Softex) e é baseado também em alguns critérios usados pelo CMMI. O MPT é voltado para a melhoria das áreas de teste de software de empresas de qualquer porte. O modelo é leve e passível de ser adotado por áreas de teste de software para apurar o seu nível de maturidade, sem com isso onerar os seus processos anteriormente implementados.

O modelo criado pela Associação Latino Americana de Teste de Software (ALATS) e pela Sociedade Núcleo de Apoio à Produção e a Exportação de Software (RIOSOFT) tem o objetivo de manter a compatibilidade com o MPS.BR e com o CMMI e permitir que empresas que implementaram o MPS e o CMMI na sua área de desenvolvimento, possam, com um pequeno esforço adicional, também aumentar e comprovar o nível de maturidade da sua área de teste de software.

O MPT.BR é um modelo que atende as empresas desenvolvedoras de software que possuem uma estrutura menor.

2.3.4.1 Níveis de Maturidade

O MPT.BR está subdividido em 7 níveis de maturidade. No primeiro nível (nível 1) a organização precisa implantar apenas a área de processo de Gerência de Projetos de Teste (GPT). Entende-se que empresas que tenham uma equipe de teste de software a princípio estarão no nível 0, embora possam ter práticas que caracterizem outros níveis de maturidade. Desta forma é importante observar que a definição de um nível de maturidade vai depender de uma avaliação das práticas usadas pela organização nos seus projetos de teste de software.

No MPT.BR cada área de processo tem a seguinte organização:

- Área de processo
- Práticas específicas

- Objetivos genéricos
- Práticas genéricas

A Tabela 3 apresenta as áreas de processo dos níveis de maturidade do modelo MPT:

Tabela 3 - Melhoria de Processo de Teste Brasileiro MPT.BR - v.2.2

Nível 1	Gerência de Projetos de Teste - GPT
Nível 2	Gerência de Requisitos de Teste - GRT
Nível 3	Aquisição – AQU (opcional)
	Gerência de Configuração – GCO
	Garantia da Qualidade - GQA
	Medição - MED
Nível 4	Gerência de Recursos Humanos - GRH
	Gerência de Reutilização - GRU (opcional)
Nível 5	Desenvolvimento de Requisitos - DRE
	Integração do Produto - ITP
	Validação - VAL (opcional)
	Verificação - VER
Nível 6	Análise de Decisão e Resolução - ADR
	Desenvolvimento para Reutilização - DRU(opcional)
	Gerência de Riscos - GRI
Nível 7	Análise de Causas e Resolução de Problemas

A seguir, os níveis do MPT.BR são descritos, com ênfase no nível 1, o nível escolhido para realizar a avaliação:

2.3.4.1.1 Nível 1

O nível 1 é o primeiro nível de maturidade do MPT. Exclusivamente no MPT existe um nível que contempla apenas Gerência de Projeto de Teste. Ao final da implantação deste nível a organização deve ser capaz de gerenciar seus projetos de teste de software, de acordo com os requisitos exigidos neste nível de maturidade. Evidentemente, a gerência de projetos de teste deverá evoluir à medida que a organização alcance níveis mais elevados de maturidade.

Para esta implementação é muito importante que a empresa utilize projetos de teste de software paralelos aos projetos de desenvolvimento de software. Ou seja, ao iniciar um projeto de desenvolvimento de software, a organização deverá ao mesmo tempo iniciar um projeto de teste de software, de forma a que os dois projetos possam caminhar de forma

paralela e integrada. Cada projeto deverá ter um gerente ou líder de projeto formalmente constituído.

O nível 1 exige a seguinte área de processo: Gerência de Projetos de Teste de Software (GPT), descrita a seguir:

Gerência de Projetos de Teste de Software (GPT)

Gerência de Projetos de Teste de Software é a única área de processo do primeiro nível do MPT e foca em gerência de projetos. Segundo o PMBOK (2004), um projeto consiste em um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo, e gerenciamento de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender aos seus requisitos.

Algumas atividades executadas nesta área de processo envolvem o seguinte:

- O desenvolvimento do Plano de Teste;
- A interação com todos os envolvidos no projeto de teste, inclusive os envolvidos com o projeto de desenvolvimento;
- O comprometimento dos interessados (stakeholders) no Plano de Teste, ou seja, a equipe de teste e demais profissionais, tais como desenvolvedores e usuários/clientes;
- O monitoramento e o controle do Plano de Teste durante toda a evolução do projeto de teste e até a sua conclusão.

A elaboração do Plano de Teste deverá ter início após o recebimento dos requisitos do negócio. Isso deve ser feito em comum acordo com a equipe do projeto de desenvolvimento, pois poderão existir requisitos específicos do projeto de teste, embora, na maior parte das situações, os requisitos de teste sejam os mesmos dos requisitos de desenvolvimento.

O processo Gerência de Projetos de Teste de Software possui 17 práticas específicas, segundo ALATS (2010), apresentadas a seguir:

Práticas específicas

O processo possui 17 práticas específicas utilizadas na avaliação dos projetos da organização no nível 1. Segue abaixo descrição de cada uma delas:

1) GPT1 - Definir o escopo do trabalho para o projeto

Em linhas gerais, o escopo geral do projeto de teste é testar o software que está sendo desenvolvido. O escopo do projeto deve ser definido e descrito em linhas gerais, e uma das melhores formas de definir o escopo é através da EAP (estrutura analítica do projeto), que é uma estrutura utilizada para desmembrar as fases de um projeto.

2) GPT2 - Estabelecer estimativas para o tamanho das tarefas e dos produtos de trabalho do projeto de teste utilizando métodos apropriados

O objetivo dessa prática é estabelecer e manter estimativas para os artefatos e para as tarefas do projeto de teste.

Devem ser feitas estimativas para produtos como Plano de Teste e Casos de Teste e devem ser realizadas algumas tarefas como gerar massa de teste, preparar ambientes de teste , executar casos de teste, dentre outros.

3) GPT3 - Definir as fases do ciclo de vida do projeto de teste

O ciclo de vida do projeto é formado por um conjunto de fases, e a definição das fases permite estabelecer alguns pontos de controle do projeto, onde alguns produtos poderão ser entregues ou produzidos

4) GPT4 - Estimar o esforço e o custo para a execução das tarefas e dos produtos de trabalho com base em dados históricos ou referências técnicas

As estimativas de esforço e custo são, normalmente, baseadas nos resultados de análises utilizando modelos e/ou dados históricos aplicados ao tamanho, atividades e outros parâmetros de planejamento.

5) GPT5 - Estabelecer e manter o orçamento e o cronograma do projeto de teste, incluindo marcos e/ou pontos de controle

O orçamento e o cronograma do projeto de teste devem ser estabelecidos com base nas estimativas de esforço e custo.

6) GPT6 - Determinar e documentar os riscos do projeto de teste, assim como seu impacto, probabilidade de ocorrência e prioridade de tratamento

Os riscos do projeto de teste devem ser identificados e analisados de tal forma que

não interfiram no planejamento e na continuidade do projeto.

7) GPT7 - Planejar os recursos humanos para o projeto considerando o perfil e a proficiência necessários para executá-lo

O conhecimento necessário para a evolução do projeto requer treinamento do pessoal envolvido no projeto, como também a contratação de pessoal com o perfil necessário e conhecimento em ferramentas e tecnologias que serão necessárias.

8) GPT8 - Planejar as tarefas, os recursos (não humanos) e o ambiente de trabalho necessário para executar o projeto de teste

Todos os recursos devem ser planejados. Entende-se por recursos, tudo aquilo que envolve o ambiente de teste, tais como, força de trabalho (que serão tratados em outra prática específica), equipamentos, ferramentas de automação, metodologias, etc. O ambiente de teste é diferente do ambiente de desenvolvimento e é aconselhável que seja semelhante ao ambiente de produção.

9) GPT9 - Identificar e planejar os artefatos e dados relevantes do projeto de teste quanto à forma de coleta, armazenamento e distribuição.

Os artefatos e dados criados pelo projeto de teste deverão estar armazenados de forma segura e confiável com um controle de versões.

10) GPT10 - Estabelecer os planos para a execução do projeto de teste e reunir no Plano de Teste

Todos os planos do projeto (cronograma, escopo) devem ser integrados ao Plano de Teste.

11) GPT11 - Avaliar a viabilidade de atingir as metas do projeto de teste, considerando as restrições e os recursos disponíveis, fazendo, se necessário, ajustes pertinentes

Deve-se fazer um estudo de viabilidade para a execução do projeto de teste de software Esta prática deve ser executada antes do início do projeto e deve ser o seu ponto de partida

12) GPT12 - Fazer a revisão do Plano de Teste com todos os interessados e obter o compromisso com o mesmo

Reunião de início de projeto e aprovação do plano do projeto por todos os presentes.

13) GPT13 - Monitorar o progresso do projeto com relação ao estabelecido no Plano de Teste e documentar os resultados

O plano de teste deverá ser monitorado durante todo o ciclo de vida do projeto de teste.

14) GPT14 - Gerenciar o envolvimento das partes interessadas (stakeholders) no projeto de teste

Os técnicos e não técnicos envolvidos no projeto devem ser identificados para a execução de cada uma das fases do ciclo de vida do projeto de teste.

15) GPT15 - Executar revisões em marcos do projeto e conforme estabelecido no planejamento

Reuniões de acompanhamento devem ser realizadas em marcos definidos no cronograma do projeto que devem estar em sintonia com o seu ciclo de vida. Deve ser feito o registro dos problemas identificados através da monitoração do projeto e esse registro deve ser controlado até a sua efetiva conclusão.

16) GPT16 - Estabelecer os registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, assim como tratar os mesmos com as partes interessadas

Deve ser feito o registro dos problemas identificados através da monitoração do projeto e esse registro deve ser controlado até a sua efetiva conclusão.

17) GPT17 - Estabelecer ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas, assim como implementar e acompanhar até a sua conclusão

Ao identificar um problema do projeto, deve ser feito o registro e o seu acompanhamento através de ações corretivas.

2.3.4.1.2 Nível 2

Segundo ALATS (2010), o nível 2 é composto pelo processo do nível 1, acrescido do processo Gerência de Requisitos de Teste (GRT).

O processo possui 5 práticas específicas:

1) GRT1 – Obter o entendimento dos requisitos de software e definir os requisitos de teste
2) GRT2 – Aprovar e obter o comprometimento com os requisitos de teste utilizando critérios objetivos
3) GRT3 – Estabelecer e manter a rastreabilidade bidirecional entre os requisitos e artefatos de teste
4) GRT4 – Realizar revisões em planos e produtos de trabalho do projeto e corrigir inconsistências identificadas em relação aos requisitos
5) GRT5 - Gerenciar as alterações dos requisitos no projeto de teste.

2.3.4.1.3 Nível 3

De acordo com ALATS (2010), o nível 3 é composto pelos processos do nível 1 e 2, acrescido dos processos: Aquisição – AQT (implantação opcional), Gerência de Configuração – GCT, Garantia da Qualidade – GQT e Medição - MET.

O processo Gerência de Configuração possui 6 práticas específicas:

1) GCT1 – Estabelecer e manter um Processo de Gerência de Configuração
2) GCT2 – Identificar os itens de configuração
3) GCT3 – Colocar os itens de configuração, sujeitos a um controle formal, sob linha básica (baseline)
4) GCT4 – Registrar e disponibilizar ao longo do tempo a situação dos itens de configuração e baselines
5) GCT5 - Controlar e disponibilizar as modificações em itens de configuração
6) GCT6 – Realizar objetivamente auditorias de configuração para assegurar que as baselines e os itens de configuração estão íntegros, completos e consistentes

O processo Garantia da Qualidade possui 4 práticas específicas:

1) GQT1 – Avaliar objetivamente a aderência dos produtos de trabalho aos padrões, procedimentos e requisitos aplicáveis, antes dos produtos serem entregues ao cliente e em marcos predefinidos ao longo do ciclo de vida do projeto de teste
2) GQT2 – Avaliar objetivamente a aderência dos processos executados às descrições de processo, padrões e procedimentos
3) GQT3 – Identificar, registrar e comunicar os problemas e as não-conformidades
4) GQT4 – Estabelecer e acompanhar ações corretivas para não-conformidades até as suas efetivas conclusões. Quando necessário, realizar o escalonamento das ações corretivas para níveis superiores, de forma a garantir sua solução.

O processo Medição possui práticas 7 específicas

1) MET1 – Estabelecer e manter os objetivos de medição a partir dos objetivos da organização e das necessidades de informação de processos técnicos e gerenciais
2) MET2 – Identificar e/ou definir, priorizar, documentar, revisar e atualizar um conjunto adequado de medidas, orientado pelos objetivos de medição
3) MET3 – Especificar os procedimentos para a coleta e o armazenamento das medidas
4) MET4 - Especificar os procedimentos para o cálculo e a análise da medição realizada
5) MET5 – Coletar e analisar os dados requeridos
6) MET6 – Armazenar os dados e os resultados de análises
7) MET7 - Usar as informações produzidas para apoiar decisões e para fornecer uma base objetiva para comunicação aos interessados

2.3.4.1.4 Nível 4

Segundo ALATS (2010), o nível 4 é composto pelos processos do nível 1,2 e 3,

acrescido dos processos: Gerência de Recursos Humanos - GRH, Gerência de Reutilização - GRU (opcional), Gerência de Riscos – GRI.

2.3.4.1.5 Nível 5

Segundo ALATS (2010), o nível 5 é composto pelos processos do nível 1,2 ,3 e 4, acrescido dos processos de Verificação – VER e Validação – VAL.

3 METODOLOGIA

Metodologia da Pesquisa, para Rodrigues (2005) consiste em estudar e avaliar os vários métodos disponíveis e suas utilizações. Corresponde a um conjunto de procedimentos que auxiliam na obtenção do conhecimento.

A classificação dos tipos de pesquisa varia de acordo com o enfoque dado, segundo interesses, condições, campos, objetivos, etc. Cabe ao pesquisador a escolha do método que melhor se aplique.

3.1 Métodos de pesquisa

A metodologia adotada nesse trabalho, quanto aos objetivos se caracteriza como pesquisa descritiva de natureza exploratória. De acordo com Gil (2002) a pesquisa descritiva tem como objetivo primordial à descrição das características de determinadas populações ou fenômenos. Uma de suas características está na utilização de técnicas padronizadas de coleta de dados, tais como o questionário e a observação sistemática.

Do ponto de vista dos procedimentos técnicos, a pesquisa pode ser classificada como Pesquisa-Ação. A pesquisa-ação, segundo Pimenta (2005) tem por pressuposto que os sujeitos que nela se envolvem compõem um grupo com objetivos e metas comuns, interessados em um problema que emerge num dado contexto no qual atuam desempenhando papéis diversos.

3.2 Técnicas da pesquisa

As técnicas, segundo Cervo e Bervian (2002) são o conjunto das diversas etapas ou passos que devem ser dados para a realização da pesquisa.

A técnica utilizada para a coleta de dados escolhida neste trabalho é o questionário. Cervo e Bervian (2002) definem questionário como um meio de obter respostas às questões por uma fórmula que o próprio informante preenche e é a forma mais usada para coletar dados, pois possibilita medir com melhor exatidão o que se deseja.

O tipo de questionário a ser utilizado é o questionário com perguntas fechadas. Rodrigues (2005) trata do questionário fechado como sendo aquelas em que o informante deve responder a pergunta através de respostas pré-definidas e aponta que este tipo de pergunta facilita sobremaneira o trabalho de tabulação dos resultados, embora restrinja a liberdade das respostas, tendo em vista o caráter objetivo do modo de questionamento.

3.3 Desenho da pesquisa

A Figura 6 ilustra uma visão esquemática da metodologia de construção do trabalho.

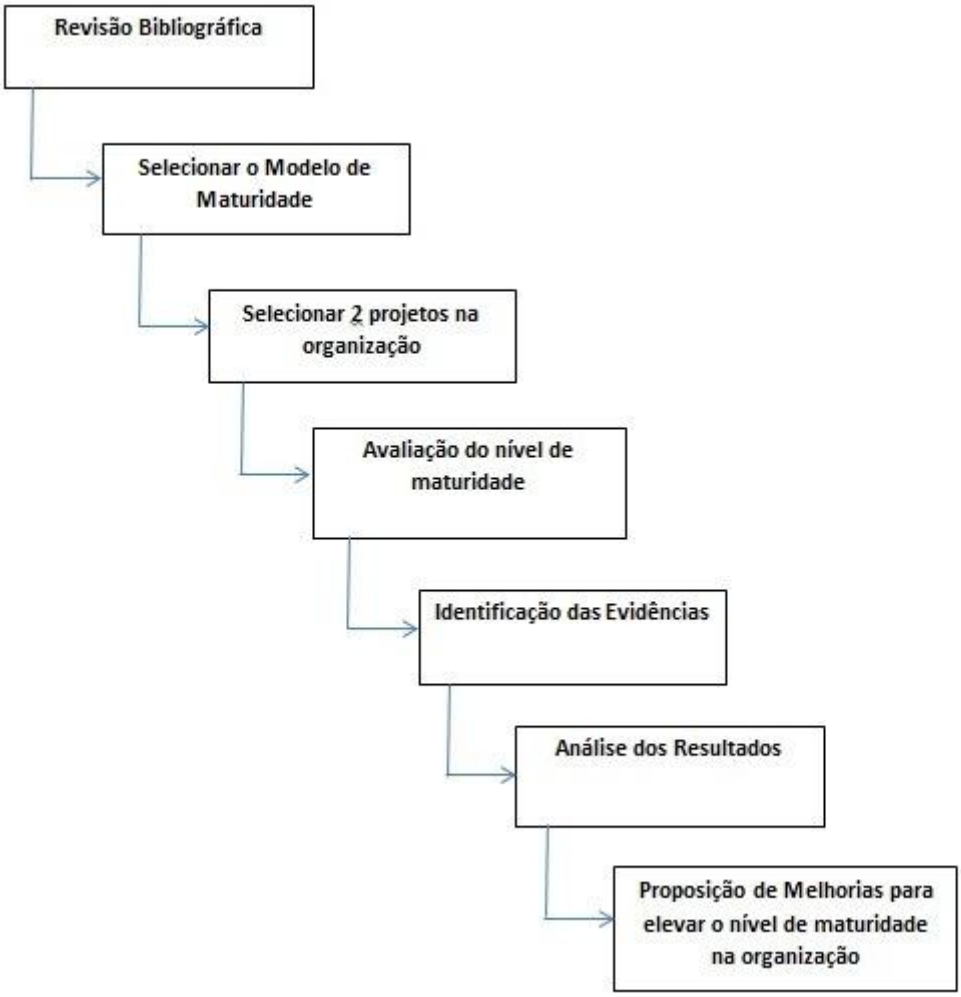


Figura 6 - Desenho da Pesquisa

Conforme observado na figura 6, a pesquisa foi realizada nas seguintes etapas: revisão bibliográfica (Testes de Software e Modelos de Maturidade), Escolha da

Metodologia da pesquisa, Formulação do questionário, descrição dos Projetos e Aplicação da metodologia proposta/Análise e interpretação do questionário.

A revisão bibliográfica segundo Cruz e Ribeiro (2003) pode visar um levantamento dos trabalhos realizados anteriormente sobre o mesmo tema estudado no momento, pode identificar e selecionar os métodos e técnicas a serem utilizados, além de fornecer subsídios para a redação da introdução e revisão da literatura do projeto ou trabalho. No presente trabalho constitui no levantamento de dados obtidos por outros pesquisadores da área de Teste de Software e Modelos de Maturidade e para isso foram consultados livros e artigos que tratam assuntos relacionados, no intuito de esclarecer mais a respeito dessa área da Engenharia de Software que é considerada uma área nova.

A seleção do modelo de maturidade foi feita levando em conta a organização e os projetos escolhidos. O nível do modelo escolhido foi o que melhor se adaptou as condições da organização.

Para avaliar o nível de maturidade, a avaliação realizada foi composta de um questionário utilizado para coleta de dados na organização, que é um documento oficial do modelo denominado “Teste de GAP Analise nível 1” que contém 20 perguntas, tendo Sim e Não como opções de resposta, e como documento auxiliar, uma planilha também foi utilizada, tendo como base, as 17 práticas específicas e os produtos típicos/evidências do nível 1 do modelo escolhido.

A última etapa corresponde à análise dos resultados do questionário e das práticas específicas da organização, com a proposta das práticas que precisam ser realizadas na organização para que o processo seja aderente ao nível 1do modelo e assim eleve seu nível de maturidade.

4 RESULTADOS E DISCUSSÃO

4.1 Caracterização da organização

Maximiano (1992) descreve uma organização como sendo uma combinação de esforços individuais que tem por finalidade realizar propósitos coletivos. Por meio de uma organização torna-se possível perseguir e alcançar objetivos que seriam inatingíveis para uma pessoa.

O alvo deste estudo são as práticas de uma organização de pequeno porte, da área tecnológica, fundada em 2003 devido à crescente demanda na área. A organização atua no ramo de tecnologia da informação.

A sede da empresa está situada na cidade de Lavras – MG, conta com uma filial na cidade de Belo Horizonte e atua no desenvolvimento e customização de software voltado para organizações públicas e privadas de médio e grande porte.

A empresa conta atualmente em seu quadro de funcionários com 6 profissionais graduados em Ciência da Computação e cerca de 10 estagiários dos cursos de Ciência da Computação e Sistemas de Informação das faculdades de Lavras- MG.

4.2 Processo de Teste de Software Atual

A organização possui uma equipe de teste que realiza testes de softwares em desenvolvimento ou já desenvolvidos, implementados ou não dentro da organização, sempre visando gerar um produto de qualidade.

A primeira versão do processo de teste proposta por SILVA (2006) baseada nas normas IEEE 829 e ISO/IEC 12119 sofreu algumas alterações práticas e foi adaptada para a versão mostrada na Figura 7, que é a versão utilizada na maioria dos projetos.

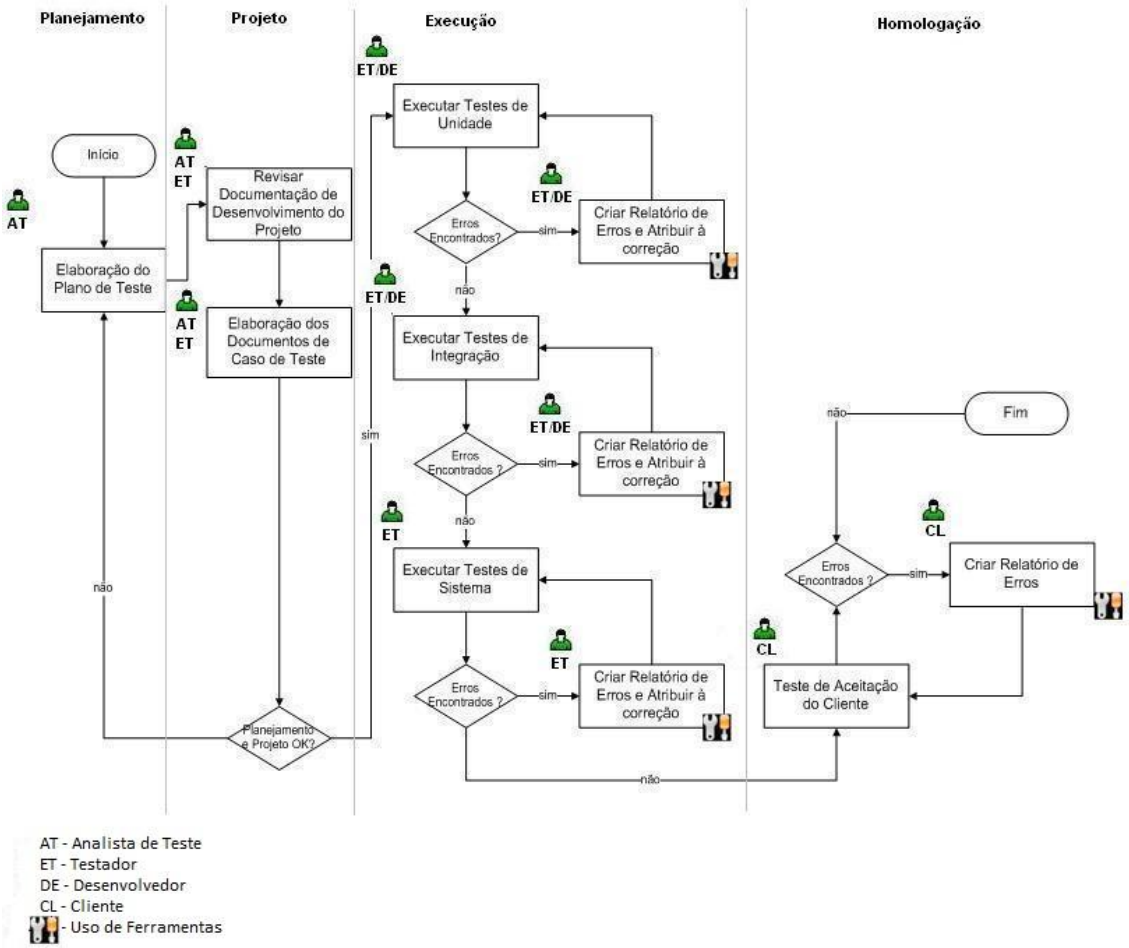


Figura 7 - Etapas do processo de teste

O processo possui quatro etapas: Planejamento, Projeto, Execução e Homologação.

1 – Planejamento

Na etapa de Planejamento, o Plano de Teste é elaborado pelo Analista de Teste. Essa etapa ocorre antes do início do desenvolvimento do sistema.

2 – Projeto

Na etapa de Projeto, a Documentação de Desenvolvimento do Projeto é revisada e são elaborados os Documentos de Casos de Teste, onde na organização em questão, cada caso de teste é baseado em um requisito pertencente ao Documento de Requisito.

O Documento de Caso de Teste serve como um checklist dos itens a serem testados

em cada requisito no que diz respeito à interface, implementação e banco de dados.

Essa etapa ocorre antes e durante o desenvolvimento do sistema.

3 – Execução

A etapa de Execução é dividida em 3 fases: Testes de Unidade, Testes de Integração e Testes de Sistema.

- Testes de Unidade

A execução dos Testes de Unidade é realizada em duas etapas:

- A primeira etapa é realizada pelo desenvolvedor, que após o término de um requisito, realiza o teste tendo como base o Documento de Caso de Teste criado pela equipe de teste.
- Finalizado o teste do desenvolvedor, o Documento de Caso de Teste é executado pelo testador.

Importante esclarecer que na literatura, encontramos definições para unidade como sendo uma função, um procedimento, uma classe, um método, e no processo da organização, cada unidade é um requisito, uma funcionalidade.

Se algum erro ou inconformidade for encontrado, um relatório detalhado é criado e enviado para o desenvolvedor utilizando a ferramenta Mantis, como mostrado na figura 8.

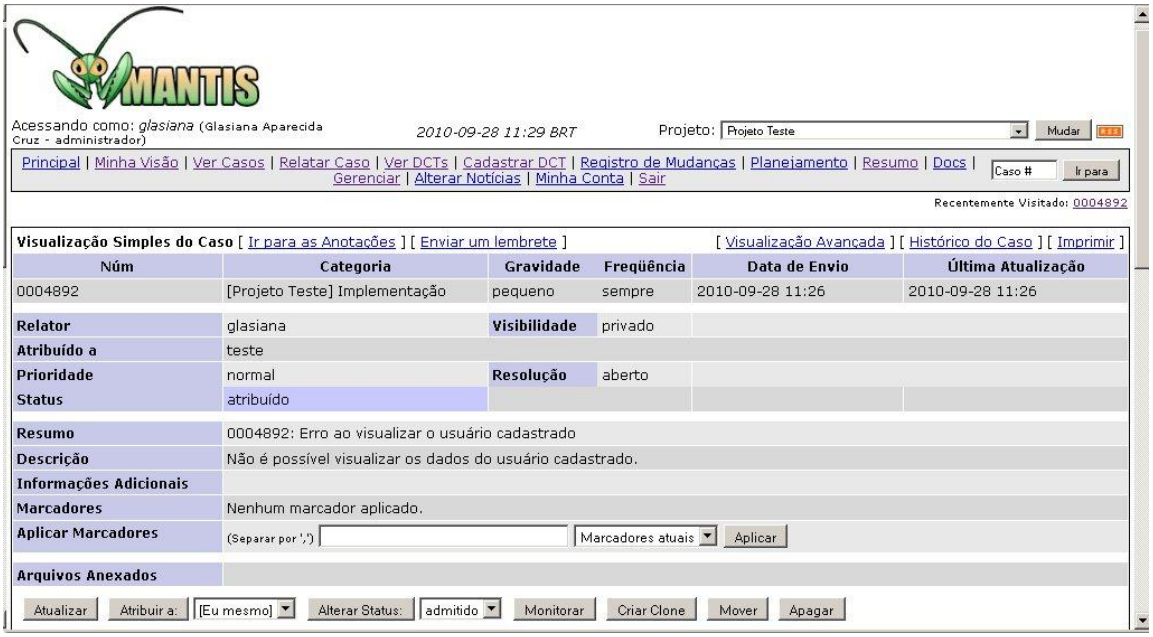


Figura 8 – Ferramenta de gerenciamento de erros usada na organização: Mantis

Mantis é uma ferramenta *Open Source* cujo principal objetivo é registrar e acompanhar o andamento de um projeto, desde o seu nascimento até o seu fechamento. Neste cenário, o ciclo de vida gerenciado pelo Mantis inicia-se quando um caso é registrado, sendo que caso pode ser um erro ou uma melhoria no projeto; as fases seguintes são empregadas para a confirmação, correção, revisão e o fechamento do caso. Cada caso recebe um número seqüencial único a fim de identificá-lo durante as operações de consulta, relatórios e modificações. Além disso, diversas informações são agregadas ao caso relatado durante as fases que compõem o seu ciclo de vida, como por exemplo, status, prioridade, severidade, comentários, anexos, etc.

- Testes de Integração

A execução dos Testes de Integração é feita da mesma forma que a execução dos Testes de Unidade. Quando funcionalidades mais complexas forem implementadas, os requisitos são testados de forma integrada para garantir o funcionamento,

A execução dos Testes de Integração é realizada em duas etapas:

- A primeira etapa é realizada pelo desenvolvedor, que após o término de um requisito, realiza o teste tendo como base o Documento de Caso de Teste criado pela equipe de teste.

- Finalizado o teste do desenvolvedor, o Documento de Caso de Teste é executado pelo testador.

Qualquer inconformidade é relatada para o desenvolvedor através da ferramenta Mantis.

- Testes de Sistema

Ao término do desenvolvimento de todos os requisitos, é realizado o Teste de Sistema, que é o teste de todas as funcionalidades do sistema integradas.

3 – Homologação

Na etapa de Homologação, o cliente realiza o teste de aceitação, e os problemas encontrados são relatados, ao desenvolvimento para serem corrigidos.

Para o cliente em questão, é utilizado um documento de aceitação das funcionalidades do sistema. Esse documento contempla os passos para a execução de cada funcionalidade, e o cliente deve executar o documento e aceitar, ou rejeitar.

Seguindo todas essas etapas, constatou que o produto chegava ao cliente com menos erros e pronto para ser utilizado.

A equipe de teste utiliza a ferramenta XPlanner, para controlar o tempo que era gasto para cada atividade do processo de teste, como mostrado nas Figuras 9 e 10:

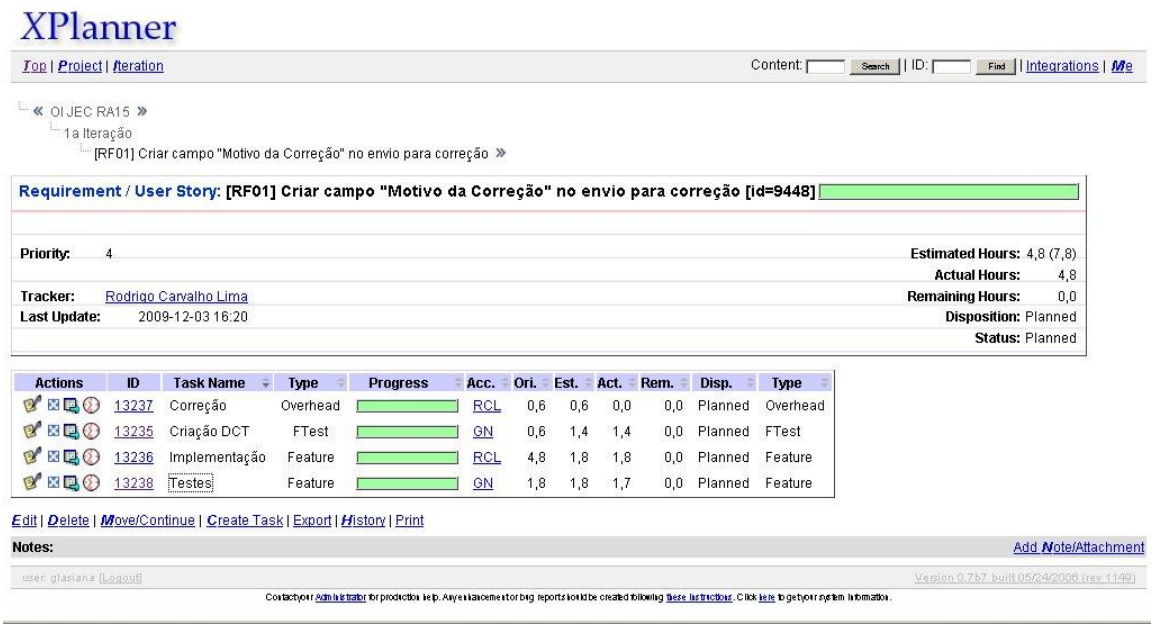


Figura 9 – Ferramenta de acompanhamento da evolução do projeto: relatório de um requisito (XPlanner)

Xplanner é uma ferramenta *Open Source* de Gerenciamento de Projetos, que possui diversas funcionalidades. Nos projetos da organização, é utilizado para lançamento dos tempos gastos nas atividades.

As tarefas (Correção, Criação DCT, Implementação e Testes) são divididas por cada responsável. É possível acompanhar o andamento de cada tarefa pelo campo Progresso e acompanhar o tempo gasto em cada tarefa pelo campo Act, como mostra a figura 10

Ao iniciar uma tarefa, o responsável por ela, preenche o campo Star Time, e ao fim da tarefa ele preenche o campo End Time e a atualiza (Update), registrando assim o tempo que ele gastou na tarefa, conforma a Figura 10:

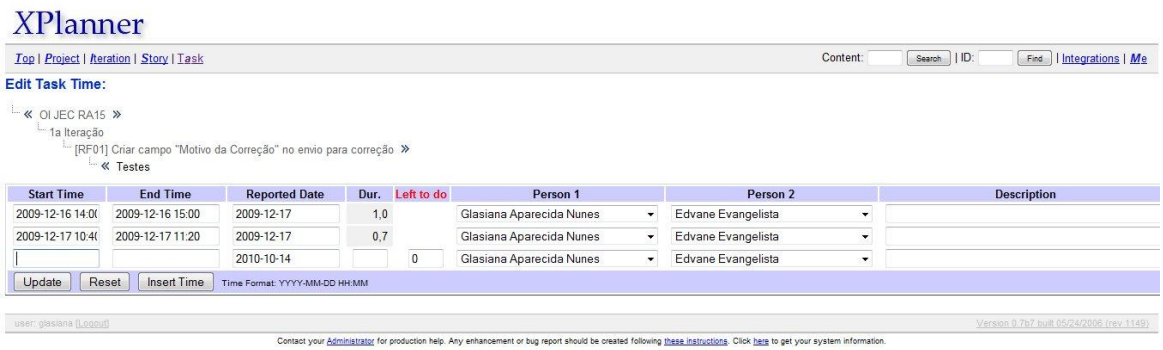


Figura 10 - Tela para preenchimento dos tempos na ferramenta XPlanner

Os documentos pertencentes aos projetos são arquivados em uma ferramenta de controle de versões para que todos os participantes dos projetos possam ter acesso e atualizar os documentos de forma organizada.

4.3 Seleção do modelo de maturidade para realização das avaliações

A organização possui iniciativas para implantação de melhorias nos seus processos e utiliza o modelo MPS.BR como padrão de desenvolvimento para os seus projetos.

Melhorar a execução dos testes através de modelos pode aumentar a maturidade nos processos e com isso ter um processo controlado onde as estimativas tendem a ser mais confiáveis, chegando a resultados mais perto do esperado.

O modelo selecionado para a avaliação foi o MPT.BR por ser um modelo brasileiro baseado no MPS.BR que se adapta bem a áreas de teste de software de empresas pequenas, como é o caso da organização descrita.

A semelhança das áreas do modelo com as áreas de processo usadas no MPS.BR , torna mais fácil de serem implementadas, pois já faz parte da cultura da organização caso ela seja usuária do MPT.BR.

O nível 1 do MPT.BR, escolhido para realizar a avaliação, favorece áreas de teste pequenas, ou que procuram um resultado a curto prazo.

4.4 Descrições dos projetos selecionados na organização estudada

Dois projetos da organização foram observados, denominados neste trabalho como “Projeto 1” e “Projeto 2”.

A avaliação foi realizada no processo de teste da organização executado em dois projetos, devido a algumas diferenças entre eles, como descrito abaixo:

4.4.1 Projeto 1

O projeto 1 é um sistema web, desenvolvido na linguagem de programação Java e que tem como banco de dados SQLServer.

Trata-se de um sistema desenvolvido para atender as solicitações de lista telefônica por parte de um cliente interno, além de manter o controle efetivo sobre as listas telefônicas entregues aos clientes. As informações da entrega de lista telefônica são digitalizadas e incluídas no sistema como comprovante de entrega das solicitações.

Depois de acordadas com o cliente as especificações do sistema, foi realizado por alguns participantes da organização, um estudo dessas especificações, e a partir delas foi confeccionado para o projeto um documento de requisitos bem detalhado, seguindo um padrão já existente na organização, onde os requisitos são divididos em funcionais e não funcionais e organizados em seções de acordo com seus atributos, sendo também atribuído a cada requisito um identificador único.

Também é padrão do documento, que os requisitos sejam organizados por sua prioridade em Alta (requisito sem o qual o sistema não entra em funcionamento), Média (requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória) e Baixa (requisito que não compromete as funcionalidades básicas do sistema). Os requisitos devem ser documentados de forma não ambígua, completa e consistente, podendo ser compreendidos, verificados, validados, modificados e rastreados.

Um protótipo do sistema foi o artifício utilizado pela equipe para garantir um entendimento comum entre cliente e equipe de desenvolvimento sobre os requisitos do sistema.

Considerado médio, o projeto contou com a participação de seis profissionais, um gerente do projeto, três desenvolvedores e dois testadores. As tarefas foram divididas utilizando o critério de maior experiência e conhecimento da tecnologia a ser utilizada.

4.4.2 Projeto 2

O projeto 2 é um sistema web, desenvolvido nas linguagens de programação ASP e Visual Basic e que tem como banco de dados SQLServer.

É um sistema desenvolvido para efetuar a análise dos documentos solicitados aos fornecedores envolvidos na Responsabilidade Solidária Recursos Humanos e Subsidiária, de acordo com cada segmento, visando manter o padrão de análise para o cálculo do risco,

caso ocorra.

O sistema já foi implementado e já é utilizado pelo cliente, o que será avaliado será um novo módulo do sistema, contendo requisitos adicionais (RA's). Para esse novo módulo, depois de acordadas com o cliente as novas especificações do sistema, foi confeccionado um documento de requisitos bem detalhado, seguindo um padrão já existente na organização, onde os requisitos são divididos em funcionais e não funcionais e organizados em seções de acordo com seus atributos, sendo também atribuído a cada requisito um identificador único.

Também é padrão do documento, que os requisitos sejam organizados por sua prioridade em Alta (requisito sem o qual o sistema não entra em funcionamento), Média (requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória) e Baixa (requisito que não compromete as funcionalidades básicas do sistema). Os requisitos devem ser documentados de forma não ambígua, completa e consistente, podendo ser compreendidos, verificados, validados, modificados e rastreados.

Um protótipo dos novos requisitos foi o artifício utilizado pela equipe para garantir um entendimento comum entre cliente e equipe de desenvolvimento sobre os requisitos do sistema.

Considerado pequeno, o módulo do projeto a ser desenvolvido contou com a participação de quatro profissionais, um gerente do projeto, dois desenvolvedores e um testador. As tarefas foram divididas utilizando o critério de maior experiência e conhecimento da tecnologia a ser utilizada.

4.5 Apresentação dos procedimentos técnicos para realização da avaliação

A avaliação proposta neste trabalho constituiu dos seguintes passos:

- Passo 1 : avaliação do nível de maturidade

A avaliação do nível 1 de maturidade é realizado através de um questionário apresentado no Apêndice A, um documento oficial do modelo, denominado Teste de GAP Análise nível 1, que serve para que cada empresa avalie se o seu nível de maturidade corresponde ao nível 1 do modelo. Ele contém vinte perguntas objetivas. A empresa respondendo SIM para todas as perguntas já é aderente ao nível 1. Esse teste constata apenas o nível da organização, mas não o seu nível de qualidade. Para isso, são usadas as

evidências das práticas específicas da área de processo do nível 1 do MPT.BR.

- Passo 2: identificação de evidências

Para uma avaliação mais profunda, foi utilizada também uma planilha com as evidências (17 práticas específicas e seus produtos típicos), apresentada no Apêndice B, que servem como indicadores que visam comprovar a adoção das práticas exigidas pelo modelo para a obtenção dos resultados esperados. Com o uso dessa planilha, fica mais fácil saber quais práticas específicas a empresa deve realizar para atingir o nível 1 do MPT.

O modelo MPT.BR não possui um modelo de avaliação particular, como o MPS.BR, por exemplo, mas o método utilizado para realizar a avaliação, que é de autoria própria do autor deste trabalho, utiliza a mesma escala utilizada no MPS.BR que caracteriza o grau de implementação de um resultado esperado do processo e de um resultado esperado de atributo do processo nos projetos, apresentadas na Tabela 4 :

Tabela 4- Escala de avaliação do grau de implementação dos indicadores (Fonte: MPS.BR – Guia de Avaliação (2009)

Grau de Implementação	Caracterização
Totalmente implementado (T)	- O indicador direto está presente e é julgado adequado - Existe pelo menos um indicador indireto e/ou afirmação confirmando a implementação - Não foi notado nenhum ponto fraco substancial
Largamente implementado (L)	- O indicador direto está presente e é julgado adequado - Existe pelo menos um indicador indireto e/ou afirmação confirmando a implementação - Foi notado um ou mais pontos fracos substanciais
Parcialmente implementado (P)	- O indicador direto não está presente ou é julgado inadequado - Artefatos/afirmações sugerem que alguns aspectos do resultado esperado estão implementados

	- Pontos fracos foram documentados
Não implementado (N)	- Qualquer situação diferente das acima
Não avaliado (NA)	- O projeto não está na fase de desenvolvimento que permite atender ao resultado ou não faz parte do escopo do projeto atender ao resultado.

- Passo 3: análise dos resultados da avaliação

Os resultados foram analisados com base nas respostas do questionário e com base no preenchimento da planilha que contem as evidências da área de processo do nível 1 comparadas às evidências encontradas na organização, através do detalhamento dos produtos típicos das práticas específicas da área de processo do nível 1 do modelo. Esse detalhamento consiste em apontar as práticas adotadas e as práticas não adotadas na organização

- Passo 4: proposição de melhorias para elevar o nível de maturidade

A proposição de melhorias teve como base os resultados da avaliação, apontando as práticas não adotadas e sugerindo a aplicação dessas práticas no processo de teste da empresa, pois é o que o modelo define como ideal para ser feito pela empresa para que ela se aproxime do nível 1 e consequentemente melhore a qualidade no seu processo e no seu produto final.

4.6 Resultados da avaliação da execução do processo de teste da organização nos projetos selecionados

A organização já possui uma cultura de testes, com um processo definido, e seus projetos já tiveram melhorias na qualidade desde o início da implantação dos testes como sendo um projeto e tendo uma equipe para realizá-los. Evangelista (2009), no seu trabalho de análise no processo de teste da organização, concluiu que o processo de teste da organização encontra-se estável, é previsível, que está em melhoria contínua, e gerou melhorias visíveis aos desenvolvedores e clientes, melhorando a qualidade do software e o

relacionamento entre os clientes e a empresa.

De acordo com a avaliação realizada através do questionário e da planilha, a organização não se encontra no nível 1 por não responder afirmativamente a todas as questões, mas realiza algumas atividades importantes percebidas no projeto 1 e no projeto 2, listadas a seguir:

- A organização possui uma área específica para executar, exclusivamente os testes;
- O teste é tratado como um projeto, com começo, meio, fim e objetivos bem definidos;
- O projeto de teste inicia junto com o projeto de desenvolvimento, pois dessa forma o custo de correções do projeto é reduzido significativamente, pois as correções realizadas no início são bem mais baratas que nos momentos finais do desenvolvimento do projeto;
- O escopo do projeto de teste é definido através de uma lista de requisitos, com todo projeto listado, separado por tarefas e responsáveis. Os projetos podem ser documentados/definidos através de um documento de requisitos, com uma descrição detalhada e objetiva de todos os requisitos funcionais e não funcionais do projeto ou através de Casos de Uso, bem descritos e detalhados;
- A organização possui base histórica para realizar estimativas, pois já trabalha com projetos similares há algum tempo. A organização procura documentar todo o esforço gasto em seus projetos, e assim poder usá-los como dados históricos para estimativas e também para realizar comparações e estudos;
- O ciclo de vida dos projetos de teste são definidos de acordo com o ciclo de vida dos projetos de desenvolvimento;
- As estimativas do esforço necessário para execução dos testes também são baseadas em dados históricos da organização;
- Os projetos possuem cronograma e esforço definidos pelo gerente do projeto, que utiliza de conhecimento prático de estimativas. Eles são documentados e toda equipe tem acesso a eles. A Figura 11 representa o cronograma de um projeto na organização:

Projeto 1

Data de início do projeto:Seg 10/05/10
Data de término do projeto:Sex 06/08/10

Dados da tarefa

Id	Nome da tarefa	Trabalho	Data de início	Data de término	Nome do recurso
1	**Projeto 1**	150 hrs	Seg 10/05/10	Sex 06/08/10	
2	**Prospecção**	7 hrs	Seg 10/05/10	Sex 21/05/10	
3	Levantamento de Requisitos	7 hrs	Seg 10/05/10	Qui 20/05/10	André Ribeiro[10%]
4	Fim Prospecção	0 hrs	Qui 20/05/10	Sex 21/05/10	
5	**Análise, Projeto e Planejamento**	8 hrs	Sex 21/05/10	Ter 22/06/10	
6	Elaborar Documento de Requisitos	4 hrs	Sex 21/05/10	Qua 26/05/10	André Ribeiro[20%]
7	Análise do Documento de Requisitos	2 hrs	Qua 26/05/10	Qua 26/05/10	André Ribeiro
8	Estimar esforço	2 hrs	Ter 22/06/10	Ter 22/06/10	André Ribeiro
9	Fim de Análise, Projeto e Planejamento	0 hrs	Ter 22/06/10	Ter 22/06/10	
10	**Cenário de Testes**	7 hrs	Qua 30/06/10	Qui 01/07/10	
11	Criar cenário de testes UAT e DCT	7 hrs	Qua 30/06/10	Qui 01/07/10	Glasiana[50%]
12	Fim de Cenário de Testes	0 hrs	Qui 01/07/10	Qui 01/07/10	
13	**Desenvolvimento**	108 hrs	Seg 10/05/10	Sex 06/08/10	
14	**Implementação**	90 hrs	Seg 05/07/10	Qua 04/08/10	
15	[RF01] Gerar Relatório de Produtividade do Usuário	45 hrs	Seg 05/07/10	Ter 20/07/10	Fernando Araújo[50%]
16	[RF02] Gerar Relatório de Lotes Suspensos por Usuário	45 hrs	Ter 20/07/10	Qua 04/08/10	Fernando Araújo[50%]
17	**Testes**	18 hrs	Ter 20/07/10	Sex 06/08/10	
18	[RF01] Gerar Relatório de Produtividade do Usuário	9 hrs	Ter 20/07/10	Qui 22/07/10	Glasiana[50%]
19	[RF02] Gerar Relatório de Lotes Suspensos por Usuário	9 hrs	Qua 04/08/10	Sex 06/08/10	Glasiana[50%]
20	Fim do Desenvolvimento	0 hrs	Seg 10/05/10	Seg 10/05/10	
21	**Fechamento**	10 hrs	Seg 10/05/10	Qua 12/05/10	
22	Elaborar Plano de Implantação	1 hr	Seg 10/05/10	Seg 10/05/10	André Ribeiro[50%]
23	Preparar o Ambiente	2 hrs	Seg 10/05/10	Seg 10/05/10	André Ribeiro[50%]
24	Implantação do Sistema em Homologação	1 hr	Seg 10/05/10	Seg 10/05/10	André Ribeiro[50%]

Figura 11 – Cronograma elaborado na organização

- As tarefas são divididas através do perfil técnico da equipe. A gerência da organização conhece o perfil técnico de cada profissional da equipe através da lista de recursos humanos da organização que possui informações sobre as capacitações dos colaboradores.
- As tarefas, os recursos e o ambiente de trabalho necessário para executar o projeto de teste são planejados. Esses planejamentos se baseiam principalmente em dados históricos comparativos e na experiência do gerente de projeto, que em geral é mais experiente e já conhece o andamento de projetos do mesmo porte e mesmas características;
- Todos os documentos pertencentes aos projetos, como documentos de planejamento ou de execução são armazenados de forma correta e segura em

uma ferramenta de controle de versão, com permissões de acordo com as responsabilidades de cada profissional participante do projeto;

- A organização realiza um estudo de viabilidade. A viabilidade é analisada com base em dados históricos de projetos anteriores e baseados na experiência do gerente de projetos que procuram avaliar parâmetros como: se o sistema se adequa aos objetivos gerais da empresa, se é possível implementar o sistema com a tecnologia atual e dentro das restrições definidas dentro do custo e prazo, dentre outros;
- Todos os documentos pertencentes aos projetos passam por revisões, para o conteúdo do projeto ser completo, claro e consistente. A figura 12 representa a documentação da revisão de um documento de requisitos da organização:

Revisões do Documento

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

Data	Versão	Descrição	Autor
27/05/2008	1.0	Criação do documento de requisitos	André Barros
27/05/2008	1.1	Atualizado com as solicitações do cliente	André Barros
27/05/2008	1.2	Revisão	Leonardo
27/05/2008	1.3	Adição do fluxo de solicitação proposto	André Barros
29/05/2008	1.4	Revisão	Leonardo
29/05/2008	1.5	Inclusão dos requisitos referentes às ações sofridas pela solicitação	André Barros
30/05/2008	1.5	Revisão após comentários de Lucia Barros e inclusão do RF07	Leonardo
02/06/2008	1.6	Revisão	Glasiana Cruz
02/06/2008	1.6	Acréscimo de RF08 e RF09	Leonardo
02/06/2008	1.6	Revisão	Glasiana Cruz
05/06/2008	1.7	Atualizado com as solicitações do cliente	André Barros
09/06/2008	1.9	Alterações visando a entrada do Fornecedor como usuário direto do sistema	André Barros
12/06/2008	2.0	Atualizado com as solicitações do cliente	André Barros
18/06/2008	2.0	Revisão	Glasiana Cruz

Figura 12 – Controle de Revisões de um Documento na organização

- A organização se adequa de acordo com o perfil técnico da equipe. Se

necessário treinamentos são realizados para a equipe ou alocação de pessoal para que as tarefas sejam cumpridas;

- Os problemas encontrados durante o projeto são documentados e monitorados até a sua conclusão. Para auxiliar no monitoramento dos problemas, a organização utiliza uma ferramenta de gerenciamento de erros, como mostrado na Figura 13. Características são relacionadas aos problemas relatados para facilitar na documentação e no monitoramento de cada caso cadastrado.

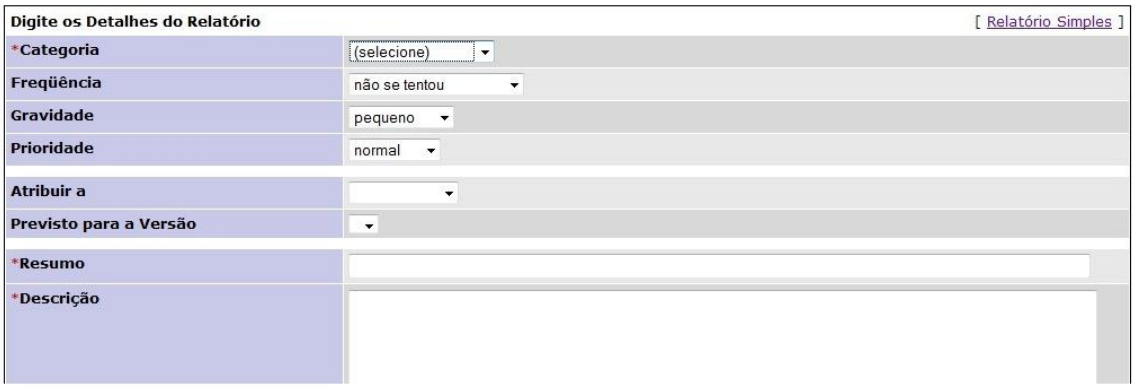
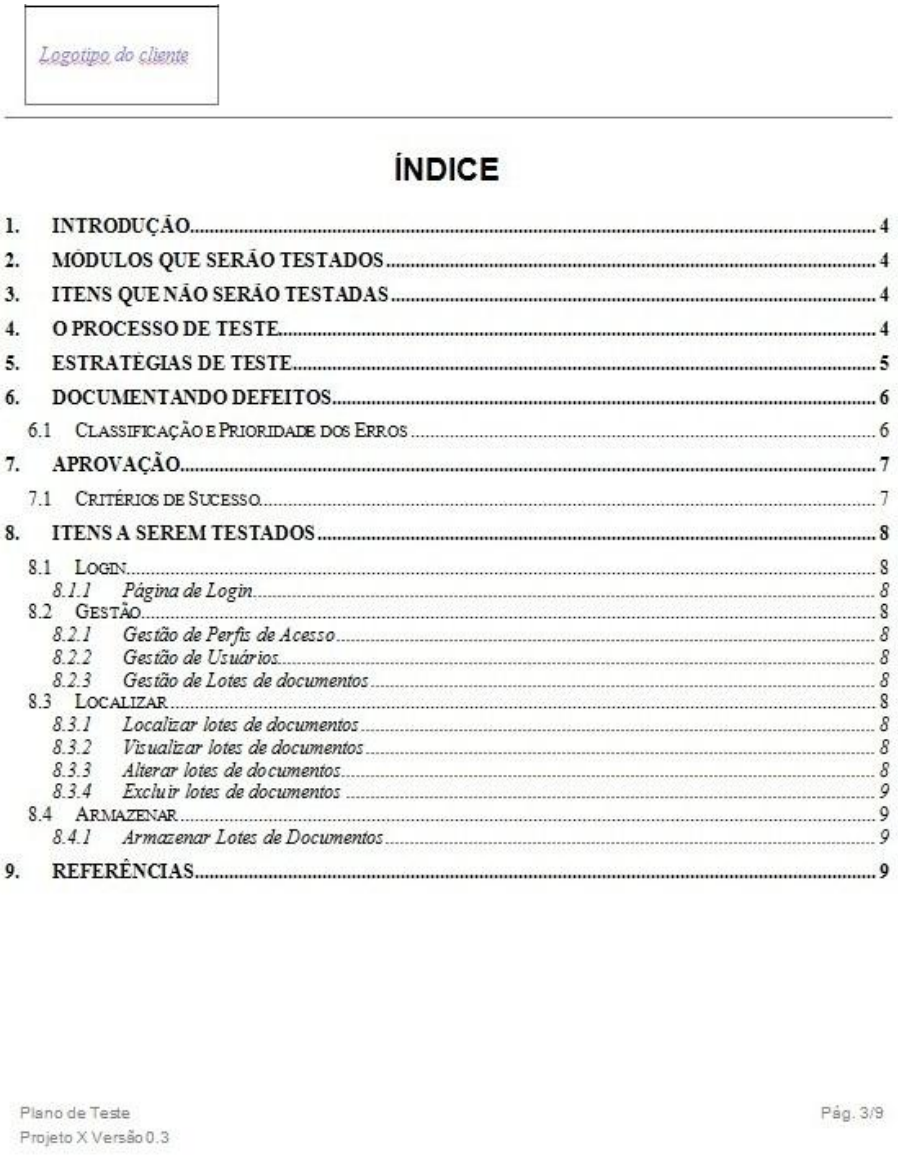


Figura 13 – Detalhes de um relatório de um problema a ser relatado na ferramenta Mantis

Mesmo a organização possuindo um processo de testes que realiza várias atividades importantes, através dos resultados da avaliação realizada, nota-se que ela ainda é falha em alguns aspectos descritos a seguir:

- A organização não realiza nenhum tipo de análise de riscos em seus projetos, e consequentemente não existe nenhum tipo de documentação sobre riscos, falhas, problemas dos projetos. Como não existe análise de riscos, também não existe nenhum tratamento dos riscos, como planos de ações corretivas do projeto. Não foi observado nos projetos avaliados grandes problemas devido à falta dessa análise, mas outros projetos já tiveram alguns problemas onde as soluções poderiam ter sido melhores e mais rápidas, se os riscos fossem conhecidos;
- O Plano de Teste utilizado nos projetos não reúne todos os planos para a execução do projeto. Informações importantes para o andamento do projeto como cronograma, prazos, não são contempladas no plano de teste, o tornando incompleto. A seguir, a Figura 14 apresenta o índice do plano de

testes utilizado na organização, com todos os itens que são contemplados no documento e que são importantes para o início do projeto como itens que serão testados, itens que não serão testados, estratégia de teste, gerenciamento de defeitos, e claramente é percebido que não contempla o cronograma, os prazos e outros planos importantes para o andamento do projeto:



Logotipo do cliente

ÍNDICE

Figura 14 – Itens contemplados no Plano de Teste da organização

- O monitoramento e a documentação das mudanças do plano de teste e de

outros documentos também devem ser realizados durante todo o projeto, regularmente ou em marcos definidos. O Plano de teste como é elaborado hoje, só é utilizado no início do projeto, e por não conter informações sobre o andamento e qualquer mudança, ele se torna incompleto. Todas as mudanças devem ser documentadas para que possam ser monitoradas de maneira correta. Foi observado que o projeto 1, por ser um projeto novo e maior, apresentou mais mudanças por parte do cliente durante o desenvolvimento que o projeto 2, e a falta de documentação dessas mudanças acarretou em atrasos e outros desvios no projeto. Já o projeto 2, por ser um módulo de um sistema já em uso, não apresentou mudanças significativas que afetassem o andamento do projeto.

4.7 Propostas de melhorias para o processo de teste da organização

As melhorias apresentadas estão segmentadas em duas etapas:

Etapa 1 – realizar análise de riscos:

No processo realizado nos projetos descritos não consta de uma análise de riscos, e com isso não é possível prever situações que podem interferir no planejamento e na continuidade do projeto. A realização dessas práticas é importante para poder prever problemas futuros e estar preparados para resolvê-los.

Analisar os riscos é o processo de avaliar ameaças, controles e vulnerabilidades. A análise procura identificar esses riscos, estimando sua severidade e montando auditorias/reuniões para mapear os possíveis riscos encontrados. Vulnerabilidades que podem se materializar em riscos são procuradas e documentadas.

Os testadores devem tentar cobrir as partes mais importantes do software, aquelas nas quais se concentrarão os maiores riscos para o projeto. Essas partes mais importantes devem ter mais prioridade na elaboração do plano de testes.

Bastos *et al* (2007), sugere que a análise de riscos, realizada por um auditor ou pelo gerente de testes, inicie pela identificação dos potenciais riscos ao processo de teste relacionados aos itens apresentados a seguir:

- Orçamento: verificar se o orçamento é insuficiente para a execução completa do projeto;

- Qualificação da equipe técnica de teste: avaliar se a equipe é capacitada para o trabalho exigido;
- Ambiente de teste: avaliar se o ambiente de testes está o mais próximo possível do ambiente de produção;
- Ferramentas: verificar se a organização possui as ferramentas que são necessárias em alguns tipos de testes;
- Metodologias: analisar se a organização possui metodologia adequada e condizente com o processo de desenvolvimento;
- Cronograma: verificar se as tarefas vão cumprir os prazos estabelecidos;
- Documentação adequada: verificar se a equipe de testes tem uma metodologia que permita guardar a documentação para uso futuro;
- Novas tecnologias: o uso de tecnologias ainda não dominadas pela equipe de teste pode ser um importante fator de risco, o que pode ser contornado com treinamento.

Depois de realizada a identificação dos riscos, é necessário classificá-los e criar um controle que evite a ocorrência desses riscos.

A frequência da ocorrência dos riscos é feita, na maior parte das vezes, por estimativas. Se não for possível testar todas as funcionalidades críticas, deverá ser feita uma análise de cada funcionalidade para saber qual pode ter mais erros, usando estimativas ou experiência profissional da equipe.

Etapa 2 – melhoria na documentação e monitoramento do projeto:

O Plano de Teste é um documento que deve reunir o máximo de informações necessárias para o planejamento de execução dos testes.

O Plano de Teste elaborado na organização não contempla os outros planos, como exibido na Figura 14. Sua elaboração deve ser de forma completa, contendo todas as informações para o andamento do projeto, reunindo outros planos que possam estar separados, como plano de escopo e cronograma, exibido na Figura 11.

O projeto deve ser monitorado com relação ao estabelecido no Plano de Teste e este deve ser monitorado durante todo o ciclo de vida do projeto de teste. Um das maneiras de monitorar o projeto é através de reuniões de acompanhamento e monitoração onde cada um dos itens do projeto é avaliado.

Toda mudança no decorrer do projeto deve ser documentada. Para um bom andamento, os documentos devem estar sempre atualizados com todas as informações úteis ao projeto. Alterações no plano de teste podem vir a ser feitas tendo em vista mudanças registradas nas reuniões de monitoramento.

Impactos das propostas de melhoria no processo de teste da organização

De acordo com as melhorias propostas, a sua implementação não gerará alterações na estrutura do processo de teste utilizado pela organização. Todas as etapas continuariam sendo executadas normalmente. Entretanto, três atividades seriam adicionadas.

A primeira delas seria a realização de uma análise de riscos, organizando uma lista de riscos do projeto de teste, identificando e analisando de tal forma que eles não interfiram no planejamento e na continuidade do projeto.

A segunda seria a elaboração de forma mais completa do Plano de Teste, ou seja, a elaboração de um plano de teste contendo outros planos.

A terceira atividade a ser adicionada seria o monitoramento do plano de teste que pode ser realizado através de reuniões de acompanhamento com a elaboração de atas das reuniões que demostrem que os itens relevantes do projeto em execução foram monitorados. Além disso, todas as mudanças devem ser documentadas para que também possam ser monitoradas de maneira correta.

4.7.1 Considerações finais sobre a avaliação

Os projetos poderiam ter melhor qualidade no processo se seguissem principalmente as 2 práticas do nível 1 descritas a seguir:

- Uma prática do modelo que os projetos da organização não realizou, principalmente no projeto 1, foi a falta de documentação das mudanças durante o projeto. Esses problemas não causaram um atraso significativo no projeto, mas geraram um desgaste por parte dos envolvidos e poderiam colocar em risco a qualidade final do produto.
- Outra prática que os projetos da organização não possuem é a análise riscos. Um projeto maior, com um maior tempo de duração tem seus riscos aumentados e é importante planejar as soluções para os problemas que

possam vir a ocorrer.

Nos projetos descritos, a previsão de eventos de risco dependeu da habilidade dos colabores da organização.

Pode ser listada como dificuldades encontradas na realização da avaliação a falta de documentação de algumas atividades realizadas, ou seja, as atividades poderiam até ser realizadas, mas se não fossem documentadas, não teria um histórico para provar.

5 CONCLUSÕES

O modelo selecionado para realizar a avaliação foi o MPT.BR por ser um modelo brasileiro baseado no MPS.BR, que é o utilizado pela organização como padrão de desenvolvimento para os seus projetos. Além disso, o MPT.BR se adapta bem a áreas de teste de software de empresas pequenas, como é o caso da organização descrita.

O diagnóstico foi feito pela avaliação do processo de teste da organização através do questionário e das práticas da organização, com base nas especificações do nível 1 do modelo MPT.BR. Os resultados demonstraram que a organização não se encontra no nível 1, mas tem um processo de teste com um certo nível de maturidade e que para elevar sua maturidade precisa adotar algumas práticas para estar aderente ao nível desejado e ter seu processo melhorado.

O estudo apontou práticas importantes não realizadas na organização que se aplicadas poderiam diminuir significativamente alguns problemas no andamento dos projetos, tanto para equipe de teste quanto para equipe de desenvolvimento.

Para elevar o nível de maturidade da organização foram propostas a realização de uma análise de riscos, a elaboração de um plano de teste contendo outros planos e o monitoramento do progresso em relação ao plano de teste e a documentação de todas as possíveis mudanças.

As propostas de melhoria não servem somente para a organização ser avaliada no nível 1 do MPT.BR, mas também para ter seu processo melhorado e controlado.

A meta da organização é estar em melhoria contínua, investindo na capacidade profissional dos empregados. Um modelo serve como referência na busca por essa melhoria, e o nível 1 do modelo estudado se apresentou simples e direto nos objetivos.

O trabalho mostra que para uma empresa que não possui uma cultura em testes e pretende investir, começar pelo nível1do MPT.BR pode ser vantajoso, pois oferece resultados a curto prazo e se adequa facilmente a pequenas empresas.

Trabalhos futuros podem ser propostos diante dos resultados obtidos:

- Proposta de adoção do nível 2 do mesmo modelo na empresa depois que todas as práticas do nível 1 forem adotadas. O trabalho seria a adoção das práticas do nível 2 no dia a dia da organização e os impactos desse trabalho no processo.

- Proposta de adoção do nível 1 em uma empresa que não possui nenhum teste definido, organizando uma equipe e propondo um processo compatível com o nível do modelo.
- Criação, baseada no estudo dos modelos existentes, de uma nova proposta de melhoria para teste de software,
- Estudo e aplicação de outros modelos em outras organizações.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ALATS, 2010 - Associação Latino Americana de Teste de Software (ALATS) e Sociedade Núcleo de Apoio a Produção e a Exportação de Software. **MPT.BR - Melhoria de Processo de teste: Guia de Implementação** - Parte 1 - Nível 1. Versão 2.2, Marco, 2010. Disponível: http://www.alats.org.br/. Acessado em 12/10/2010.

ALATS, 2010 - Associação Latino Americana de Teste de Software (ALATS) e Sociedade Núcleo de Apoio a Produção e a Exportação de Software. **MPT.BR - Melhoria de Processo de teste: Guia de Implementação** - Parte 1 - Nível 2. Versão 1.1, Marco, 2010. Disponível: http://www.alats.org.br/. Acessado em 20/10/2010.

ALATS, 2010 - Associação Latino Americana de Teste de Software (ALATS) e Sociedade Núcleo de Apoio a Produção e a Exportação de Software. **MPT - Melhoria de Processo de teste: Guia de Implementação** - Parte 1 - Nível 3. Versão 0.0. Marco, 2010. Disponível: http://www.alats.org.br/. Acessado em 20/10/2010.

ALMEIDA, R **Automatizar ou não automatizar?**, Test White Paper T&M Testes de Software Ltda, IBQTS, 2007.

ALVES, G. de S.. **Modelo de Maturidade em Testes com Foco em Ambientes de Testes Heterogêneos**. Dissertação de Mestrado, Cin- UFPE, 2007.

BARTIÉ, A. **Garantia da qualidade de software: adquirindo maturidade organizacional**. Rio de Janeiro: Campus, 2002.

BASTOS, A.; Rios, E.; CRISTALLI, R. & Moreira, T. **Base de conhecimento em teste de software**. São Paulo, Editora Martins Fontes, 2007.

BEIZER, B. **Black-box testing** techniques for functional testing of software and systems. New York: John Wiley & Sons, 1995

BLACK, R. **Managing the Testing Process**, Second Edition. Wiley, New York, 2002.

BURNSTEIN, I., **Practical Software Testing: A Process-oriented Approach**. 2003.

Springer, New York

CERVO, A. L. e BERVIAN, P. A. **Metodologia científica**. 5a ed., São Paulo: Prentice Hall, 2002. (pag 26)

CMMI – **Guidelines for Process Integration and Product Improvement** – Mary Beth Chrissis e outros – Ed. Adisson Wesley – 2004.

COLEMAN, G., and O'CONNOR, R. **"Using grounded theory to understand software process improvement: A study of Irish software product companies, Information and Software Technology,"** (49:6), pp. 654–667, 2007.

CONTI, C., T. **Modelo de Maturidade Aplicado ao Processo de Teste de Software**. Novo Hamburgo: 2007. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Instituto de Ciências Exatas e Tecnológicas, Centro Universitário Feevale, 2007.

CORSO Ariane Louise, **AVALIAÇÃO DA APLICABILIDADE E EFICÁCIA DE UM MODELO DE MATURIDADE DE TESTE**, PIRACICABA, SP 2008 (Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.)

CRESPO, A. N. et. Al. (2004) **"Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo"**, Simpósio Brasileiro de Qualidade de Software, Maio, p.271-285.

CRISTALLI, R *et al*,. **Referência Complementar Certificação Brasileira de Teste de Software** – CBTS. 2007

CRUZ, Carla; RIBEIRO, Uirá. **Metodologia científica – teoria e prática**. Rio de Janeiro: Axcel Books do Brasil Editora, 2003.

DELGADO, S. **"The Impending Implementation of CMMI for Test Software"**

EVALUATION ENGINEERING -CHICAGO- 2007, VOL 46; NUMB 9, pages 12-17

ERICSON, T., SUBOTIC, A., URSING, S.. **TIM – A Test Improvement Model**. 2003

EVANGELISTA, A. E. **Análise do Subprocesso de Teste de Software de uma Empresa.** 2009. Artigo (Graduação em Sistemas de Informação) - Faculdade Presbiteriana Gammon – Lavras, MG.

FERREIRA, G. K. **Teste de Usabilidade.** 2002 Monografia (Graduação em Ciência da Computação) – Universidade Federal de Minas Gerais, Belo Horizonte, MG. Disponível em: http://www.slideshare.net/marceloramos/avaliao-de-usabilidade.

GAO, J.. Z.; TSAO, H. -S. J. e WU, Y. **Testing and Quality Assurance for Component Based Software**. Massachusetts: Atech House, 2003, 439p.

GOMES, N. S. **Qualidade de software – Uma necessidade**. [Acessado em 18 de março de 2009] Disponível na internet < http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf

GRAF, C.; SERRANO. A., S. **TESTE DE SOFTWARE. Revista de divulgação técnico-científica do ICPG** - Vol. 3 n. 9 - jul.-dez./2006

HARROLD. M. J**. Testing: a roadmap**. In: 22nd International Conference on Software Engineering. Junho, 2000.

HERBERT , J. S., **Teste Cooperativo de Software. Teste** (Doutorado em Ciência da Computação). Porto Alegre: Instituto de Informática, Universidade Federal do Rio Grande do Sul, 1999

HETZEL, W. **The Complete Guide to software testing**. 2 ed. QED Info Science Inc., 1988.

HORGAN, J.R., MATHUR, A.P. **Assessing testing tools in research and education**. IEEE Software, v. 9, n. 3, maio de 1992

HUMPHREY, W.S. **Managing the software process**. Addison-Wesley, 1990.

IEEE Computer Society; IEEE Std 829: Standard for Software Test Documentation; September, 1998.

ISO. **Guidelines for the application of iso 9001 to the development, supply and maintenance of software**. IS 9000-3. ISO, 1991.

JUNG, C. F. **Metodologia para pesquisa & desenvolvimento aplicada a novas tecnologias, produtos e processos.** Ed. Rio de Janeiro: Axcel Books, 2004.

KALINOWSKI, M., SANTOS, G., REINEHR, S., MONTONI, M., ROCHA, A.R., WEBER, K.C., TRAVASSOS, G.H. **"MPS.BR: Promovendo a Adoção de Boas Práticas de Engenharia de Software pela Indústria Brasileira"** In: CIbSE - XIII Congreso Iberoamericano en "Software Engineering", Universidad del Azuay, Cuenca, Ecuador, 12-16 Abril 2010.]

KOSCIANSKI, A.; SOARES, M. dos S. **Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2ª edição. Ed. São Paulo: Novatec, 2007.

MAGELA, R. **Engenharia de software aplicada: fundamentos**. 1ª edição. Ed. Rio de Janeiro: Alta Books, 2006.

MAXIMIANO, A. C. A. **Introdução a administração**. 3ª edição. Ed. São Paulo: Atlas, 1992.

MOLINARI, E. **Testes de Software: Produzindo Sistemas melhores e mais Confiáveis**. São Paulo: Érica, 2003. 228p

MOLINARI, L. **Testes Funcionais de Software**. Editora Visual Books, 2008

SOFTEX. MPS.BR - Melhoria de Processo do Software Brasileiro: Guia de avaliação. Versão 2009. Disponível em:

http://www.softex.br/mpsbr/_guias/guias/MPSBR_Guia_de_Avaliacao_2009.pdf. Último acesso em Outubro de 2009. Ocorrência: 2009c.

MYERS, G.. **The Art of Software Testing**. 2 ª ed. John Wiley, 2004.

PAULA FILHO, W. DE P. **Engenharia de Software: fundamentos, métodos e padrões**. 2a ed. Rio de Janeiro: LTC - Livros Técnicos Científicos. 2003. ISBN: 8521613393.

PERRY, W. E.; **Effective Methods for Software Testing**, 3rd edition, Wiley Publishing, Indianapolis, IN, 2006. 591p

PESSÔA, M., S., P. – **Introdução ao CMMI – Modelo Integrado de Maturidade da Capacidade de Processo** / Ed. Lavras/MG: UFLA/FAEPE, 2008

PETERS, J., F.; WITOLD, P. **Engenharia de software**. Rio de Janeiro: Campus, 2001.

PIMENTA, S. G.; **Pesquisa-ação crítico-colaborativa: construindo seu significado a partir de experiências com a formação docente**. Educação e Pesquisa, São Paulo, v. 31, n. 3, p. 521-539, set./dez. 2005

PRESSMAN, R. S. **Software engineering - a practitioner's approach**. 5th. ed. McGraw-Hill, 2001.

PRESSMAN, R. A. **Engenharia de Software**. 6º Edição. São Paulo: McGraw-Hill, 2006

REFFSON A.; BEZERRA C.I.; COUTINHO E. **Análise da Aderência de um Processo de Teste ao TMM**. Artigo Técnico, SBTS – I Simpósio Brasileiro de Teste de Software, 2006.

RODRIGUES, M. G. V. **Metodologia da pesquisa: elaboração de projetos, trabalhos acadêmicos e dissertações em ciências militares** / Maria das Graças Villela Rodrigues. colaboração e ampliação José Fernando Chagas Madeira, Luiz Eduardo Possídio Santos, Clayton Amaral Domingues - 2. ed - Rio de Janeiro: EsAO, 2005

ROUILLER, A. C.; Vasconcelos, A. M. L. de; Maciel, T. M. de M. **Engenharia de Software** – Lavras: UFLA/FAEPE, 2004.

SANTOS, J,. M., S. **Um Framework para definição de processos de testes de software que atenda ao nível 3 do TMM-e**. Recife: s.n., 2006. 86p.

SILVA, W. M. P. **Swtest: um Processo de Teste Definido para uma Empresa de Pequeno Porte Desenvolvedora de Software**. 2006. Monografia (Graduação em Ciência da Computação) – Universidade Federal de Lavras, Lavras, MG. Disponível em: http://www.bcc.ufla.br/monografias/2006/PrimeiraTurma.html. Acessado em setembro de 2010

SOFTEX. MPS.BR - **Melhoria de Processo do Software Brasileiro: Guia Geral**: 2009. Versão de atualizada em setembro de 2009. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf. Último acesso em Outubro de 2010. Ocorrência: 2009a.

SOGETI. **Test Management Approach**. S.I.: 2007

SOMMERVILLE, I.; SAWYER, P. **"Requirements Engineering – a good practice guide".**

STAAB, T. C**. Improving the Test Process – Looking at the Test Process – Getting Started**. The Journal of Software Testing Professionals, March 2003.

SWEBOK, **Guide to the SWEBOK**. Disponível em (http://www.swebok.org). Acessado em 18/05/2010

TOSETTO, M. **Ambiente Organizacional para Teste de Software.** 2004. Monografia (Graduação em Informática – Análise de Sistemas) – Universidade do Vale do Rio dos Sinos, RS. Disponível em: http://www.unisinos.br/inf/images/stories/Inf/32tc_mauro_tosetto.pdf. Acessado em 18/05/2010.

VASCONCELOS, Alexandre Marcos Lins, ROUILLER, Ana Cristina, MACHADO, Cristina Ângela Filipak Machado, MEDEIROS, Teresa Maria Maciel. **Introdução à Engenharia de Software e à Qualidade de Software**. Editora FAEPE - UFLA, 2006

VEENENDAAL, E. **TMMi Foundation. Test Maturity Model Integration (TMMi)**. Versão 2.0. Ireland, 2009. . Disponível: http://www.tmmifoundation.org/html/resources.html. Acessado em 12/10/2010

VILAS BOAS, A. L. de C. **Qualidade e Avaliação de Produto de Software**. Ed. Lavras/MG: FAEPE/UFLA, 2007.

WEBER, K. C., ARAÚJO, E., MACHADO, C. A. F., SCALET, D., SALVIANO, C. F., da ROCHA, A. R. C., "**Modelo de Referência e Método de Avaliação para Melhoria de Processo de Software versão"** 1.0 (MR-MPS e MAMPS)" In: IV Simpósio Brasileiro de Qualidade de Software, Porto Alegre, Brasil, 2005

YIN, R. K. **Case study research: design and methods**. 2nd. ed. Thousand Oaks, 1994.

7 APÊNDICE A – Questionário

O questionário “Teste de GAP Análise – Nivel 1” serve para que a empresa avalie o nível de maturidade da sua área de teste de software em relação ao nível 1 do MPT. Esse teste constata apenas a existência das evidências, mas não o seu nível de qualidade.

O questionário é composto por vinte perguntas, tendo SIM (S) e NÃO (N) como respostas e é respondido pela autora que participou efetivamente dos dois projetos. Está disponível em http://www.alats.org.br/.

Gerência de Projetos de Teste de Software			
PREENCHIDO PELA EMPRESA			
Teste de GAP Analise nivel 1		Projeto 1	Projeto 2
O propósito da área de processo Gerência de Projetos de Teste de Software é tornar a organização capaz de gerenciar seus projetos de teste de software, de acordo com os requisitos exigidos neste nível de maturidade.			
1 - Na sua empresa tem uma área específica para executar, exclusivamente, os testes?		S	S
2 - O teste, neste caso, é tratado como um projeto?		S	S
3 - O projeto de teste inicia junto com o projeto de desenvolvimento?		S	S
4 - O escopo do projeto de teste é definido através de uma Estrutura Analítica do Projeto ou WBS - work breakdown structure ou através de algum outro modelo? Exemplo: Lista de tarefas ou pacotes de trabalho.		S	S
5 - Existe um critério técnico para estimar o tamanho do projeto de teste?		S	S
6 - Os projetos de teste têm um ciclo de vida definido formalmente?		S	S
7 - Existe um critério técnico para estimar o esforço necessário para a execução dos testes?		S	S
8 - Os projetos de teste têm um cronograma e um orçamento?		S	S
9 - É feita uma análise de riscos dos projetos de teste?		N	N
10 - Existe um critério técnico para a escolha da equipe do projeto de teste considerando o seu perfil?		S	S
11 - As tarefas, os recursos e o ambiente de trabalho necessário para executar o projeto de teste são planejados?		S	S
12 - Os documentos do projeto de teste são armazenados de forma correta e segura? Obs.: Neste nível não precisa ser uma gerência de configuração.		S	S
13 - O Plano de Teste contempla os outros planos (escopo, prazo, custos, etc.) e é compatível com o padrão PMI?		N	N
14 - Existe um estudo de viabilidade do projeto de teste?		N	N
15 - Os documentos de teste, principalmente do Plano de Teste, são revistos, ou seja, passam por revisões técnicas?		S	S
16 - O projeto de teste é monitorado regularmente?		N	N
17 - O envolvimento da equipe de projeto de teste é monitorado? Ex.: Necessidades de treinamento e contratação são definidas.		S	S
18 - Estão programadas no Plano de Teste reuniões de acompanhamento do projeto de teste?		N	N
19 - Os problemas encontrados no monitoramento do projeto são registrados num documento próprio para esse fim?		S	S
20 - Os problemas encontrados são monitorados até a sua conclusão?		S	S

8 APÊNDICE B – Planilha de apoio à avaliação baseada nas práticas do Modelo MPT.BR Nível 1

A planilha mostrada a seguir foi preenchida pela autora (analista de teste) participante dos dois projetos. Para melhor compreensão seguem algumas orientações:

- Primeira coluna: corresponde às práticas específicas da área de processo Gerência de Projetos de Teste de Software do nível 1 do MPT.BR, com evidências indicadoras do nível;
- Segunda coluna: é marcada quando a organização realiza a prática nos outros projetos;
- Terceira coluna: é marcada quando o projeto 1 realiza a prática da coluna 1 correspondente;
- Quarta coluna: é marcada quando o projeto 2 realiza a prática da coluna 1 correspondente.

Os documentos utilizados para a elaboração da planilha encontram-se disponíveis em http://www.alats.org.br/.

Gerência de Projetos de Teste de Software				
PREENCHIDO PELA EMPRESA				
Prática específica / evidências	ORG	Projeto 1	Projeto 2	Observações
O propósito da área de processo Gerência de Projetos de Teste de Software é tornar a organização capaz de gerenciar seus projetos de teste de software, de acordo com os requisitos exigidos neste nível de maturidade.Essa área possui 17 práticas específicas vistas a seguir:				
PE 1 - GPT1 – Definir o escopo do trabalho para o projeto				
Descrição do escopo do projeto, das atividades, descrição dos pacotes	X	X	X	O escopo do projeto, das atividades é definido no Plano de Testes
Definiçao e resumo da EAP (estrutura analítica do projeto)		X		Existe um documento (lista de requisitos) que organiza os requisitos/funcionalidades em partes, e essa divisão foi idêntica a do desenvolvimento, para os 2 projetos.
Lista de requisitos	X	X	X	
(T,L,P,N,NA)	P	L	P	
PE 2 - GPT2 – Estabelecer estimativas para o tamanho das tarefas e dos produtos de trabalho do projeto de teste utilizando métodos apropriados				
Medição do tamanho e complexidade das tarefas e dos produtos de trabalho	X	X	X	Estimativas com dados históricos, baseados na experiência de outros projetos já realizados
Indicadores e históricos usados nas estimativas	X	X	X	
Modelos usados para elaborar as estimativas	X	X	X	
(T,L,P,N,NA)	L	L	L	
PE 3 - GPT3 - Definir as fases do ciclo de vida do projeto de teste				
Ciclo de vida do projeto de teste de software com fases e subfases	X	X	X	O ciclo de vida do projeto de teste nos 2 projetos foi dividido de acordo com o ciclo de vida do projeto de desenvolvimento
(T,L,P,N,NA)	L	L	L	

PE 4 - GPT4 – Estimar o esforço e o custo para a execução das tarefas e dos produtos de trabalho com base em dados históricos ou referências técnicas				
Racional dos cálculos de estimativa	X	X	X	Estimativas com dados históricos, baseados na experiência de outros projetos já realizados.
Estimativa dos esforços do projeto de teste	X	X	X	
Estimativa dos custos do projeto de teste.	X	X	X	
(T,L,P,N,NA)	L	L	L	
PE 5 - GPT5 – Estabelecer e manter o orçamento e o cronograma do projeto de teste, incluindo marcos e/ou pontos de controle				
Cronograma do projeto de teste	X	X	X	cronograma construído juntamente com gerente de projeto
Dependências do cronograma do projeto de teste				
Orçamento do projeto de teste	X	X	X	em geral , o projetos de teste da organização são estimados como sendo 30% do projeto de desenvolvimento
(T,L,P,N,NA)	P	P	P	
PE 6 - GPT6 – Determinar e documentar os riscos do projeto de teste, assim como seu impacto, probabilidade de ocorrência e prioridade de tratamento				
Riscos identificados				não foi realizado para nenhum dos projetos, nenhum tipo de análise de risco
Impacto e probabilidade de ocorrência dos riscos				
Prioridade de tratamento dos riscos				
Planos de mitigação e de contingência.				
(T,L,P,N,NA)	N	N	N	
PE 7 - GPT7 – Planejar os recursos humanos para o projeto considerando o perfil e a proficiência necessários para executá-lo				
Planilha com o perfil das necessidades de recursos humanos do projeto	X	X	X	são realizados treinamentos, quando necessário, e a qualificação de cada um é conhecida através da lista dos recursos humanos da organização
Lista dos recursos humanos do projeto indicando as necessidades de contratação	X	X	X	
Qualificações do pessoal e as necessidades de treinamento.	X	X	X	
(T,L,P,N,NA)	L	L	L	

PE 8 - GPT8 – Planejar as tarefas, os recursos (não humanos) e o ambiente de trabalho necessário para executar o projeto de teste				
EAP detalhada contemplando os recursos necessários	X	X		Recursos de ambiente e equipamentos listados e documentados
Requisitos de pessoal baseados no escopo e no tamanho do projeto				
Equipamentos e ambientes de teste	X	X	X	
(T,L,P,N,NA)	P	P	P	
PE 9 - GPT9 – Identificar e planejar os artefatos e dados relevantes do projeto de teste quanto à forma de coleta, armazenamento e distribuição.				
Plano de gerência de dados	X	X	X	todos os documentos do projeto são arquivados em uma ferramenta de controle de versão, com permissões
Plano de comunicação	X	X	X	
Requisitos de privacidade e segurança dos dados.	X	X	X	
(T,L,P,N,NA)	L	L	L	
PE 10 - GPT10 – Estabelecer os planos para a execução do projeto de teste e reunir no Plano de Teste				
Plano de teste contemplando todos os outros planos.				Plano de teste não possui cronograma nem escopo integrado para nenhum dos projetos
(T,L,P,N,NA)	N	N	N	
PE 11 - GPT11 – Avaliar a viabilidade de atingir as metas do projeto de teste, considerando as restrições e os recursos disponíveis, fazendo, se necessário, ajustes pertinentes				
Análise preliminar de viabilidade	X	X	X	a análise de viabilidade é realizada baseado também em dados históricos e experiência dos colaboradores
(T,L,P,N,NA)	L	L	L	
PE 12 - GPT12 – Fazer a revisão do Plano de Teste com todos os interessados e obter o compromisso com o mesmo				
Ata de reunião de início (kick-off) com as assinaturas dos envolvidos (internos e externos);	X	X	X	Revisões dos documentos ao início do projeto de todos interessados no projeto.
Plano de envolvimento com as devidas responsabilidades.	X	X	X	
(T,L,P,N,NA)	L	L	L	

PE 13 - GPT13 –Monitorar o progresso do projeto com relação ao estabelecido no Plano de Teste e documentar os resultados				
Atas de reuniões de acompanhamento do projeto demonstrando que os itens relevantes do projeto foram monitorados				não existe um registro de mudanças, nem monitoramento durante o ciclo de vida
Registro de mudanças com a sua monitoração até a conclusão				
(T,L,P,N,NA)				
PE 14 - GPT14 – Gerenciar o envolvimento das partes interessadas (stakeholders) no projeto de teste				
Lista de todos os técnicos envolvidos no projeto	X	X	X	informações(conhecimento e responsabilidades) de todos os envolvidos no projeto documentadas, lista de recursos necessários e realização de treinamentos, caso necessite
Necessidades de técnicos em termos de contratação ou treinamento	X	X	X	
Regras e responsabilidades dos técnicos envolvidos com respeito à responsabilidade no projeto por fase do ciclo de vida e por atividade.	X	X	X	
Recursos necessários (treinamento, equipamentos, orçamento, etc.) para que os técnicos envolvidos possam desenvolver a sua atividade no projeto.	X	X	X	
(T,L,P,N,NA)	L	L	L	
PE 15 - GPT15 – Executar revisões em marcos do projeto e conforme estabelecido no planejamento				
Atas de reuniões de acompanhamento do projeto demonstrando que os itens relevantes do projeto foram monitorados				não existe um acompanhamento/monitoramento durante o projeto, nem documentação de mudanças em marcos definidos no cronograma
Registro de mudanças com a sua monitoração até a conclusão.				
(T,L,P,N,NA)	N	N	N	
PE 16 - GPT16 – Estabelecer os registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, assim como tratar os mesmos com as partes interessadas				
Registro de mudanças com a sua monitoração até a conclusão (por conclusão podemos considerar o registro para uma futura resolução) .	X	X	X	os problemas/erros são registrados através e monitorados de ferramentas
(T,L,P,N,NA)	L	L	L	
PE 17 - GPT17 – Estabelecer ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas, assim como implementar e acompanhar até a sua conclusão				
Registro de mudanças com a sua monitoração até a conclusão com o registro das ações corretivas adotadas.	X	X	X	os problemas/erros encontrados são corrigidos com o auxílio de ferramentas
(T,L,P,N,NA)	L	L	L	