Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

CARLOS MAGNO DIAS RUSSO

Implantação do processo VER e VAL de software usando MPS.BR em uma fabrica de software

Belo Horizonte
2011

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação
Especialização em Informática: Ênfase: Análise de Sistemas

Implantação do processo VER e VAL de software usando MPS.BR em uma fabrica de software

por

CARLOS MAGNO DIAS RUSSO

Monografia de Final de Curso

Prof. Antonio Mendes Ribeiro
Orientador

Belo Horizonte
2011

CARLOS MAGNO DIAS RUSSO

Implantação do processo VER e VAL de software usando MPS.BR em uma fabrica de software

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais, como atividade da disciplina Desenvolvimento de Pesquisa e Projeto de Informática II, ministrada pelo professor Antônio Mendes Ribeiro.

Área de concentração: Curso de especialização Análise de sistemas à distância

Orientador: Prof. Antonio Mendes Ribeiro

Belo Horizonte
2011

RESUMO

Com objetivo de obter o nível C do MPS.Br muitas empresas estão buscando adquirir alta qualidade em todos os seus processos. Devido a este processo, a qualidade do software esta cada dia mais em evidência seja pelo cliente que deseja um produto de qualidade, ou pela empresa que deseja possuir excelência nos seus produtos com objetivo de adquirir novos clientes a cada dia. Como ocorre em toda mudança é necessário que as organizações mudem hábitos, paradigmas e adotem processos que venham a garantir a melhoria em todas as etapas do desenvolvimento do software. Esta monografia tem como objetivo abordar a aplicação do processo de verificação e validação proposto no MPS.Br em um software de gestão hospitalar, abordando todos os processos, elaboração e execução de testes, mostrando que com planejamento e execução dos testes desde o processo inicial do projeto até a entrega do software ao cliente, com certeza irá obter bons resultados de qualitativos ao produto final.

Palavras-chaves: MPS.Br, qualidade, software, verificação, validação, processo, testes.

ABSTRACT

Aiming to achieve the C level on MPS.Br many companies are looking to get high quality in all their processes. Because the process of quality of software is becoming increasingly evident, is the customer who wants a quality product or is the company you that tries to get excellence in their products with the objective of obtain new customers every day. Like occurs with any changing process it is necessary for organizations to change habits, paradigms and adopt processes that ensure improvement in all phases of software development. This monograph aims to address the implementation of verification and validation process proposed in MPS.Br in a hospital management software, covering all processes, preparation and execution of tests showing that, with planning and execution of tests from the initial process of design to the delivery of software to the customer is possible to obtain a good quality final product.

Keywords: MPS.Br, quality, software, verification, validation, processing, testing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPU ……....Unidade central de processamento
Fhemig.....Fundação Hospitalar de Minas Gerais
IEEE .....Instituto de Engenheiros Eletricistas e Eletrônicos
CMMI.....Capability Maturity Model Integration
MPS.BR.....Melhoria de Processos do Software Brasileiro
PMI.....Project Management Institute
QAI.....Quality Assurance Institute
RUP……Rational Unified Process
SCI .....Itens de configuração de software
SEPG.....Software Engineering Process Group
SIGH.....Sistema de Gestão Hospitalar
TI.....Tecnologia da Informação
UML.....Unified Modeling Language
VER .....Verificar
VAl.....Validar

SUMÁRIO

1 Introdução

Em um ambiente cada vez mais competitivo as empresas de TI estão em busca continua em construir produtos de software com qualidade, com objetivos de se manterem vivas no mercado que cada dia esta mais disputado. As empresas procuram avaliar seus processos não só com o objetivo de obter uma avaliação que comprove seu nível de qualidade, mas também para identificar deficiências e melhorias nos seus processos, corrigindo falhas de forma a aprimorá-los continuamente (PORTO 2007). A atividade de testes não deve ser considerada como uma fase isolada em um processo de desenvolvimento, mesmo dentro de um processo interativo, devendo por isso ser integrada a cada fase de desenvolvimento (INTHURN, 2001). Para controlar o desenvolvimento surgiu a disciplina de engenharia de software. Ela compreende um conjunto de etapas que envolvem métodos, ferramentas e os procedimentos seqüência em que os métodos serão aplicados (PRESSMAN, 1995). Pensando nisso esta monografia apresenta uma proposta de melhoria de processos baseada no mps.Br (Melhoria de Processos do Software Brasileiro) aplicada ao contexto de uma empresa de TI cujo objetivo consiste em conquistar o certificado nível C do mps.Br.

Para obter o nível C é necessário que as empresas passem pela melhoria ou criação do processo de desenvolvimento do software com foco na qualidade do produto que inclui os testes desde o inicio do projeto já na fase de levantamentos de requisitos até a entrega e validação do produto final pelo cliente. É um equivoco pensar nos testes somente na fase final do projeto. A Realização da verificação e da validação poderá garantir que no final do projeto o produto estará com qualidade suficiente para deixar os usuários satisfeitos. Na verificação, as inspeções e revisões sobre os produtos gerados pelas diversas etapas do processo de desenvolvimento são conhecidas como testes estáticos. Já na validação, os processos de avaliar se o sistema atende aos requisitos do produto são conhecidos como testes dinâmicos.

São 7 os níveis de maturidade do mps.Br que ficou dividido da seguinte forma: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). Começando em ordem decrescente pelo nível G (Parcialmente Gerenciando) até o nível A, de forma que a cada nível que for atingido significa que os niveis de maturidade vão sendo acumulados.

O nível G é o primeiro nível do mps.Br que é composto pelos processos de gerência de projetos e gerência de requisitos e tem como objetivo estabelecer e manter planos que definem atividades, recursos e responsabilidade.

O nível F é o segundo nível do mps.Br é composto pelos processos de aquisição, garantia da qualidade e gerencia de configuração, gerência de portifólio de projetos e medição e tem como objetivos gerenciar a aquisição de produtos.

O nível E é o terceiro nível que é composto pelos processos de avaliação e melhoria do processo organizacional, definição do processo organizacional e gerencia de recursos humanos e reutilização.

O nível D é o quarto nível que é composto pelos processos de desenvolvimento de requisitos, integração de produto, projeto e construção de produto, validação e verificação que tem como objetivo confirmar que cada serviço e ou produto de trabalho atende os requisitos especificados.

O nível C é o quinto nível que é composto pelos processos de análise de decisão e resolução e gerência de riscos que tem como objetivo analisar possíveis decisões usando um processo formal de acordo com os critérios estabelecidos.

O nível B é o sexto nível e é composto pelos processos desempenho do processo organizacional e gerência quantitativa que tem como objetivo estabelecer e manter um entendimento quantitativo do desempenho dos processos padrão da organização para apoiar os objetivos de qualidade.

O nível A é o sétimo é ultimo nível do mps.Br e tem como objetivo implantação das inovações na organização e análise e resolução de causas de forma mensurada, melhorando os processos e as tecnologias da organização.

Os níveis de maturidade são acumulados, a cada nível atingido passa ser executados pelo nível de capacidade correspondente, por exemplo, quando a organização atingir o nível C quer dizer que os níveis superiores D, E, F e G já foram implementados e estão sendo executados.

A cada nível de maturidade atingido os processos vão sendo acumulados ou seja na passagem de um nível de maturidade superior os processos implementados devem ser executados no nível superior, para atingir o nível A todos processos passados tem que estar devidamente implementado e funcionando de acordo com o processo estabelecido.

1.1 Justificativa

Nota-se atualmente que muitas empresas de desenvolvimento de software estão em busca de desenvolver um produto de alta qualidade com baixo custo. Muitos paradigmas estão sendo quebrados, estruturas sendo modificadas, saindo da visão tradicional baseada em áreas funcionais em direção a redes de processos centrados no cliente. A competitividade no mercado de desenvolvimento de software faz com que cada dia às empresas invistam não só na qualidade dos seus produtos como também no processo de desenvolvimento como forma de aumentar a qualidade do processo e produto final. Por isso, muitas empresas de TI estão aderindo ao mps.Br que se baseia nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software.

1.2 Objetivo

O objetivo principal dessa monografia é apresentar a situação de uma empresa que esta em busca do nível C do mps.Br, onde esta sendo implantado todo o processo de testes desde o inicio do projeto até a fase final executando a verificação e a validação dos produtos.

Para tanto são listados a parte teórica que envolve os principais passos a serem implantados durante o ciclo de desenvolvimento enfatizando a fase de testes de verificação e validação dos artefatos / programas de acordo com o processo de implantação do mps.Br, até a parte pratica é apresentado um estudo de caso onde é descrito o resultado do processo de verificação e validação.

2 Referencial teórico

2.1 Processo de teste

O processo de teste de software deve-se basear em uma metodologia aderente ao processo de desenvolvimento, em pessoal técnico qualificado, em ambiente e ferramentas adequadas (MOREIRA FILHO, 2003).

***Planejamento*:** Indica a abrangência, a abordagem, os recursos e a programação da atividade de testes (PETER E PEDRYCS,2001).

Projeto: Indicam características especificas a serem testadas os casos de testes associados (PETER E PEDRYCS,2001).

Implementação: Os itens a serem testados são instalados e configurados, assim como as ferramentas e componentes de teste (PAULA FILHO,2001).

Execução: Executam-se os testes e são produzidos os relatórios correspondentes (PAULA FILHO,2001).

O processo de teste começa com testes que exercitam o conteúdo e a funcionalidade da interface visível aos usuários finais. A medida que o teste prossegue, tópicos do projeto de arquitetura e de navegação são exercitados.

2.1.1 Ciclo de vida do processo de teste

O ciclo de vida do processo de teste, é composto por diversas etapas ou fases, sendo quadro delas seqüências ou em cascata e duas paralelas (RIOS, 2007).

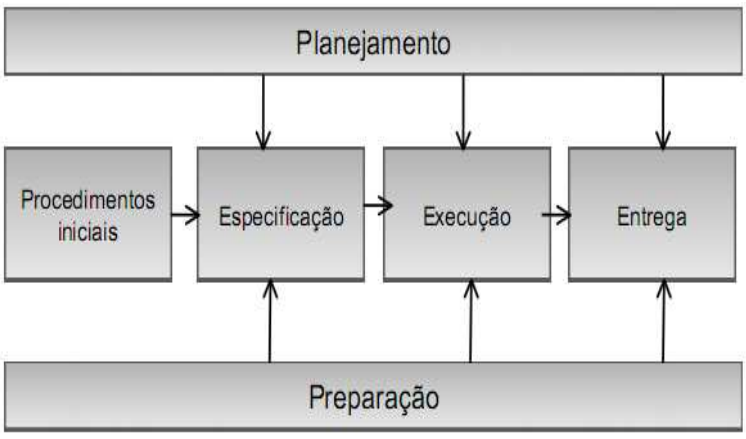

Figura 1. Ciclo de vida do processo de testes

Planejamento e a preparação acompanham todo o processo de testes.
Os Procedimentos iniciais é uma fase curta na qual é traçado um pequeno esboço do processo de teste. Especificação, execução e entrega consomem em torno de 85% de todo o processo.

2.1.2 Procedimentos iniciais

Elaborar um plano com todas as atividades principais que serão executadas, incluindo as necessidades de recursos de pessoal e de ambiente.

2.1.3 Planejamento

A atividade de planejamento tem de permanecer ativa até que o projeto seja concluído, visto que se fará necessário avaliar constantemente se os rumos do projeto estão dentro do que foi previsto e planejado.
Para cumprir esses objetivos, algumas diretrizes precisam ser consideradas:
- Testes de verificação deverão ser executados sobre os requisitos do sistema, para minimizar inconsistências, faltas e incorreções;
- Deverá ser preparada a análise de riscos do projeto de testes.

2.1.4 Preparação

O objetivo básico é preparar o ambiente de teste com equipamentos, pessoal, ferramentas de automação, hardware w software, para que os testes sejam executados corretamente.

2.1.5 Especificação

Os objetivos básicos desta etapa são os seguintes:
- Elaborar/ revisar casos de teste;
- Elaborar/ revisar roteiros de teste.

É preciso que se observe o seguinte: os casos de testes e os roteiros de teste devem ser elaborados dinamicamente durante o decorrer do projeto de teste. Isto quer dizer que

eles serão elaborados à medida que a equipe de desenvolvimento liberar módulos ou partes dos sistemas para teste.

2.1.6 Execução

Executar os testes planejados e registrar os resultados obtidos são tarefas que precisam obedecer às diretrizes a seguir:

- Os testes deverão ser executados de acordo com os casos de testes e os roteiros de teste;
- Caso seja utilizado alguma ferramenta de automatização deverá utilizar os scripts de testes;
- Os testes deverão ser executados integralmente, por regressão ou parcialmente, sempre que surgir alguma mudança de versão dos programas em teste e nos ambientes de teste preparados, desenvolvimento, testes, homologação, produção, conforme previsto no plano de teste.

2.1.7 Entrega

O projeto de teste é finalizado. Será concluída/arquivada toda a sua documentação e serão relatadas todas as ocorrências desse projeto que forem consideradas relevantes à melhoria do processo.

2.2 Conceito V de teste

O ciclo de vida de testes pressupõe que sejam realizados testes ao longo de todo o processo de desenvolvimento. Os ciclos de vida de testes e desenvolvimento são totalmente interdependentes, mas o ciclo é dependente da conclusão dos produtos das atividades do ciclo de desenvolvimento.

As atividades do ciclo de vida de testes só podem ser bem realizadas se forem executadas por um grupo formalmente definido para executar os testes. Este grupo pode ser formado internamente ou externamente.

Internamente: Equipe interna formada para executar os testes de todos os projetos de desenvolvimento de software.

Externo: Equipe externa independente especializada em executar testes de software.

O importante é que o grupo de teste se baseie em uma metodologia formal de testes.

O ciclo de vida de testes é ilustrado na figura 2. A figura mostra que os processos de desenvolvimento e teste têm inicio simultaneamente: a equipe de desenvolve o sistema inicia o processo de desenvolvimento do sistema, e a equipe responsável pelos testes começam o planejamento do processo de teste. Ambas as equipes começam no mesmo ponto usando as mesmas informações.

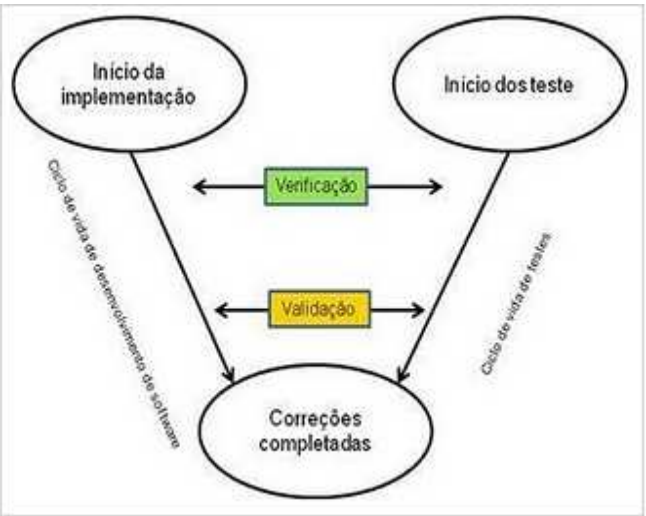

Figura 2. Conceito “V” de teste de software.
Disponível em < http://gustavoquezada.blogspot.com/2009/06/perguntas-e-respostas-parte-2.html > Acesso em 08 Dez. 2010

2.2.1 Processo de teste de software

A verificação e a validação devem ser usadas para o desenvolvimento e a manutenção de softwares. Alguns passos do processo de testes são:

Acesso ao plano de desenvolvimento:

É um pré-requisito para a construção do plano de teste. Durante este passo, os testadores verificarão se o plano de desenvolvimento esta correto. Com base nesse plano será possível estimar a quantidade de recursos necessários para testar a solução a ser implementada.

Desenvolvimento do plano de teste:

A preparação do plano de testes segue os mesmos padrões da preparação do plano de desenvolvimento, a estrutura é a mesma, mas o conteúdo irá variará em função do grau de risco associado com o software que está sendo desenvolvido.

Inspeção ou teste dos requisitos do software:

Avaliação dos requisitos do software mediante o uso da técnica de verificação, requisitos incompletos pode levar ao fracasso o desenvolvimento do software.

Inspeção ou teste do desenho do software

É utilizada a técnica de verificação para avaliar o desenho interno e externo, verificando se o desenho atende aos objetivos dos requisitos, bem como se é eficaz e eficiente para operar no hardware previsto.

Inspeção ou teste da construção do software

A partir do desenho do sistema determinará o tipo e a extensão dos testes que serão necessários. Quanto mais a construção se tornar automatizada, menos testes serão requeridos durante esta fase.

Execução dos testes

Testar o código em estado dinâmico. A abordagem, as ferramentas e os métodos especificados no plano de testes serão utilizados para validar os códigos executáveis aos requisitos do software e suas especificações de desenho.

Teste de aceitação:

Avaliar se o software está apto a ser implantado com nível de erros ainda não corrigidos. Deve-se identificar as expectativa do usuário em relação ao sistema separando o que for considerado “erro” ou “mudança” de regras ou requisitos.

Informação dos resultados dos testes:

É um processo contínuo pode ser informal ou formalizado com documentação, o mais importante é que os defeitos ou tópicos envolvidos sejam relatados as pessoas envolvidas o mais rápido possível para ser corrigido o mais breve possível diminuindo o custo do projeto.

Teste de instalação do software:

Visa verificar a interoperabilidade com o sistema operacional, como outros softwares relacionados e com os procedimentos operacionais. O resultado vai determinar se o produto está ou não em condições de ser implantado no ambiente de produção.

Teste de mudança de software

Cobre as mudanças durante o processo de implementação e aquelas que irão ocorrer após a implantação do software.

Avaliação da eficácia dos testes:

As melhorias no processo de teste podem ser verificadas com maior exatidão pela avaliação da eficácia dos testes ao término de um projeto. Deve ser realizada pelos testadores, porém envolve os desenvolvedores, usuários e outros profissionais inseridos no processo de qualidade.

2.3 Técnicas de Teste

A técnica de teste que será utilizada nos diferentes estágios são fatores importantes no processo. Por exemplo, para um teste de estresse ser executado é necessário um grande volume de dados e que este seja criado de maneira estruturada e o mais próximo possível da realidade. Já a técnica funcional visa garantir que o software foi desenvolvido de acordo com a especificação, por exemplo, o teste de regressão que visa garantir que após uma implementação ou mudança o software continua funcionando corretamente.

2.3.1 Técnica de teste Estrutural

O Objetivo dos testes estruturais é garantir que o produto seja estruturalmente sólido e que funcione corretamente. As técnicas para esse tipo de teste são desenhadas não para garantir que o sistema seja funcionalmente correto, e sim para que ele seja estruturalmente robusto.

2.3.1.1 Testes de Estresse

O Objetivo desse tipo de teste é avaliar o comportamento do software sob condições críticas, tais como restrições significativas de memória, área de disco e de CPU. O teste de estresse é realizado colocando-se software sob condições mínimas de operação (RON PATTON, 2005). Os softwares de teste de estresse visam também identificar o comportamento do software quando submetido a variados volumes de dados e acima das médias esperadas. Transações, tabelas internas, espaços em disco, saídas, capacidade do computador e interação com pessoas são aspectos que devem ser submetidos a estresse.

Os testes de estresse devem simular o mais próximo possível o ambiente de produção. Software on-line devem ser estressados com pessoas entrando as transações com volumes normais e acima dos normais. Software bach podem ser estressados com grande volume de dados. É importante incluir situações de erro nas transações submetidas nos testes.

As transações para uso em teste de estresse podem ser obtidas de uma das seguintes fontes:

- Geradores de dados de testes;
- Transações criadas pelo grupo de teste;
- Transações processadas no ambiente de produção.

Exemplos de testes de estresse:

- Determinar se foi alocado espaço em disco suficiente para a aplicação;
- Garantir que a capacidade de comunicação seja suficiente para trafegar o volume de trabalho esperado;
- Testar as condições de estresse, entrando com mais transações do que as tabelas, as filas, os dispositivos internos de armazenamento conseguem armazenar.

2.3.1.2 Testes de execução

Os testes de execução são usados para avaliar o comportamento do sistema no ambiente de produção e verificar se são atendidas as premissas de desempenho estabelecidos. Os testes de execução verificam os tempos de resposta, os tempos de processamento e o desempenho. A execução de um sistema pode ser testada por inteiro ou em partes, usando o próprio sistema ou modelo simulado.

Os testes são capazes de avaliar um simples aspecto do sistema, como, por exemplo, uma rotina crítica ou habilidade de um algoritmo satisfazer a um critério de desempenho.
Os testes de execução podem ser executados da seguinte forma:
- Usando monitores de hardware e software;
- Simulando o funcionamento de todas as partes ou apenas de uma determinada seção de um sistema, usando um modelo de simulação;
- Criando programas temporários para avaliação de desempenho.

Exemplo teste de execução:

- Calcular o tempo de processamento para transações processadas pela aplicação;
- Confirmar que o hardware e o software selecionados fornecem capacidade de processamento adequada;
- Usar monitores de software para determinar se o código de um programa é umpregado com eficácia.

2.3.1.3 Testes de recuperação

A recuperação é a capacidade de reiniciar operações após a perda da integridade de uma aplicação. Esse processo geralmente requer que se volte a um ponto do processamento no qual a integridade do sistema é garantida e então se reprocessam as transações até o ponto da falha.
Responsável por garantir a continuidade das operações após um desastre, o teste de recuperação não só verifica o processo de recuperação como também a eficácia das partes componentes do processo.
Os testes de recuperação podem ser executados da seguinte forma:
- Mantendo back-up dos dados
- Armazendo os dados de backup em locais diferentes e seguro;
- Documentando os procedimentos de recuperação
- Nomear pessoas encarregadas de recuperar os dados
- Utilizando ferramentas de recuperação

Exemplos de testes de recuperação

- Induzir um dos programas do sistema a falhar;
- A recuperação poderia ser conduzida a partir de um ponto conhecido, no qual garante a integridade dos dados. A tarefa seria recuperar esses dados a partir de uma posição mais adiante e garantir que o backup dos dados se adequasse ao processo de recuperação.

2.3.1.4 Testes de Operação

Após os testes, a aplicação deve ser integrada com o ambiente operacional. Nesse momento, o sistema será executado usando os operadores, os procedimentos e a documentação. Os testes de operações são usados para verificar, antes da entrada da aplicação em produção real.
Os testes de operações são usados principalmente para estabelecer se o sistema é executável durante a operação normal. Os objetivos são:
- Determinar se a documentação da operação esta completa;
- Testar se os operadores, usando a documentação preparada, conseguem efetivamente operar o sistema.

Exemplos de testes de operadores

- Determinar se as instruções para a operação foram preparadas e documentadas de acordo com os procedimentos da produção e se os operadores foram treinados para situações anormais.
- Verificar se a rotulagem de arquivos e os procedimentos de proteção funcionam adequadamente.

2.3.1.5 Testes de Conformidade

Os testes de conformidade verificam se a aplicação foi desenvolvida de acordo com os padrões, procedimentos. É mais importante executar os testes de conformidade durante a fase de requisitos do que nos estágios finais do ciclo de vida, porque é mais difícil corrigir aplicações quando os requisitos não estão adequadamente documentados. Os objetivos específicos são:

- Verificar se as metodologias de desenvolvimento de software e de manutenção são seguidas;
- Avaliar se a documentação do sistema de aplicação é racional e se esta completa.

2.3.1.6 Testes de Segurança

Os testes de visam descobrir defeitos difíceis de identificar. Entre os objetivos da segurança são:
- Determinar se foi realizado atenção adequada à identificação de riscos de segurança;
- Determinar se foi preparada uma definição realista das regras de acesso ao sistema e se estas foram implementadas de acordo com as definições;
- Conduzir testes racionais para garantir que as medidas de segurança tenham sido corretamente implementadas.

Exemplos de testes de segurança

- Determinar se os recursos a serem protegidos estão identificados e os acessos estão definidos para cada recurso;
- Avaliar se os procedimentos de segurança foram implementados adequadamente e funcionam de acordo com as especificações;
- Tentar o acesso não autorizado em software on-line para garantir que o sistema seja capaz de identificá-lo e impedi-lo.

2.3.2 Técnica de teste Funcional

Os testes funcionais do sistema são realizados para garantir que os requisitos e as especificações do sistema tenham sido atendidos. Entre os tipos de técnicas úteis na execução dos testes funcionais
- testes de requisitos;
- testes de regressão;
- testes de tratamento de erros;
- testes de suporte manual;
- testes de interconexão com outros softwares;
- testes de controle;
- testes de paralelos.

2.3.2.1 Testes de requisitos

Visam verificar se o sistema executa corretamente as funcionalidades e se é capaz de sustentar essa correção após sua utilização por um período de tempo contínuo

Exemplos de testes de requisitos

- Criar uma matriz de testes para provar que os requisitos do sistema como foi documentado, são os requisitos desejados pelos usuários.
- Usar um checklist preparado especificamente para aplicação, a fim de verificar sua conformidade com as políticas da organização e os regulamentos.
- Determinar se o sistema atende aos requisitos de auditoria estabelecidos pelas áreas de auditoria interna e de auditoria externa.

2.3.2.2 Testes de tratamento de erros

Determinam a capacidade do sistema de tratar apropriadamente transações incorretas. Determina se todas as condições de erro esperados são conhecidos pelo sistema.

Exemplo de testes de tratamento de erros

- Produzir um conjunto representativo de transações contendo erros e introduzi-lo no sistema para determinar se este administra os problemas;
- Entrar com dados cadastrais impróprios, para determinar o comportamento do software na gestão desses erros.

2.3.2.3 Testes de regressão

Os testes de regressão voltam a testar segmentos já testados após a implementação de uma mudança em outra parte do software. Sempre que mudanças são efetuadas no código fonte, problemas podem ocorrer em outras partes já testadas, sendo assim os objetivos da regressão são:

- Verificar se a documentação do sistema permanece atual;
- Verificar se os dados e as condições de testes permanecem atuais;
- Verificar se as funções previamente testadas continuam funcionando corretamente após a introdução de mudanças no sistema.

Exemplos de testes de regressão

- Executar os testes antes realizados com o intuito de garantir que a parte não alterada do sistema funciona de forma correta;
- Rever o manual de procedimentos já preparado para garantir que ele continua correto após a introdução de mudanças no sistema de aplicação;
- Verificar o dicionário de dados para garantir que a documentação dos dados alterados está correta.

2.3.2.4 Testes de suporte manual

Os testes manuais envolvem, a principio, a avaliação da adequação do processo e sua execução, que pode ser feita justamente com o teste normal do sistema. Os testes podem ser realizados com o pessoal envolvido preparando e entrando as transações, usando o sistema e empregando os resultados do sistema de aplicação.

Exemplos de testes de suporte manual

- Fornecer ao pessoal que entra com dados o tipo de informação que eles normalmente receberam de seus clientes, depois transforma-los para permitir a entrada no sistema automatizada.
- Fornecer aos usuários uma série de condições, depois solicitar que eles respondam quais ações são adequadas.

2.3.2.5 Testes de interconexão

A interconexão pode se dar através de dados recebidos, fornecidos ou de ambos. Estes testes são realizados como forma de garantir que a interconexão entre softwares de aplicação funcione corretamente.

Exemplos de testes de interconexão

- Desenvolver um conjunto representativo de transações de teste num software para passarem para outros softwares visando verificar a correta transferência.
- Verificar se a documentação dos softwares afetada está atualizada.

2.3.2.6 Testes de controle

Entre os controles estão a validação de dados, a integridade de arquivos, o backup, a recuperação, a documentação e outros aspectos do sistema relacionados à integridade. O teste de controle é a ferramenta de gestão necessária para assegurar que o processamento seja realizado conforme sua intenção. Os objetivos são:

- Os dados estejam completamente corretos;
- As transações sejam autorizadas;
- O processo atenda às necessidades dos usuários

Exemplos de testes de controle

- Selecionar transações de teste e verificar se o processamento para essas transações pode ser reconstituído.
- Determinar se existe segurança adequada de que o total de registros de um arquivo é igual ao registrado no arquivo de controle do sistema.

2.3.2.7 Testes paralelos

Os testes paralelos exigem que os mesmos dados de entrada rodem em duas versões da mesma aplicação, podendo ser realizados com o software inteiro ou com apenas um ou mais segmentos. Os testes paralelos possuem os seguintes objetivos:

- Assegurar que a nova versão do software de aplicação execute corretamente;
- Demonstrar consistências e inconsistências entre duas versões do mesmo software de aplicação.

Exemplo de testes paralelos

- Operar a versão nova e a antiga do software de aplicação para determinar se os resultados são conciliáveis

2.4 Atributos de qualidade

Os softwares devem atender a certos atributos de qualidade e, para isso precisamos identificar seus objetivos ou requisitos de qualidade, que complementam os requisitos funcionais, desempenho, custo e de cronograma. Os requisitos de qualidade devem ser transmitidos para a equipe de desenvolvimento e documentados da mesma maneira que os outros requisitos.

Os atributos de qualidade a serem atendidos devem também fazer parte do plano de testes.

2.4.1 Custo versus qualidade

O custo da qualidade inclui todos os custos decorrentes da busca da qualidade ou da execução das atividades relacionadas à qualidade. Os custos da qualidade podem ser

divididos em custos associados com a prevenção, revisões, técnicas formais, equipamento de testes, treinamento. Os custos de falha são aqueles que desapareceriam se nenhum defeito aparecesse antes de se entregar um produto ao cliente. Os custos com falha podem ser subdivididos em custos de falhas internas e custos de falhas externas. Os custos de falhas internas ocorrem quando detectamos um defeito no produto antes de entrar em produção. Os custos de falha interna incluem refazer, reparar, análise do modo como a falha ocorreu. Os custos de falha externa são associados com os defeitos encontrados depois que o produto foi enviado ao cliente. Exemplos de custos de falha externa são soluções das queixas, devolução e substituição do produto. Como esperado, os custos relativos para encontrar e reparar um defeito aumentam significativamente a medida que migramos dos custos de prevenção para os de detecção de falhas internas até os de falha externa.

Identificar os fatores de qualidade mais importante: Os fatores de qualidade considerados importantes para o software são classificados de acordo com a ordem de importância. Isto é mostrado na primeira coluna da tabela 1, cuja linhas já seguem essa ordem de prioridade.

Fatores	Ciclo de vida							
	Analise dos requisitos	Desenho	Codificação e testes	Testes do sistema	Operação	Revisão	Transição	Economia
Correção	x	x	x	o	o	o		Alta
Confiabilidade	x	x	x	o	o	o		Alta
Eficiência	x	x	x		o			Baixa
Integridade	x	x			o			Baixa
Usabilidade	x	x	x	o		o	o	Média
Manutenibilidade		x	x			o	o	Alta
Testabilidade		x	x	o		o	o	Alta
Flexibilidade		x	x			o	o	Média
Reusabilidade		x	x				o	Média
Interoperabilidade	x	x		o			o	Baixa

Tabela 1. Custo esperado versus custo de realização.

X – Fatores de qualidade que devem ser medidos
O – Impacto de baixa qualidade

As definições dos fatores de qualidade:

Correção: Extensão em que um programa satisfaz a suas especificações e atende aos objetivos dos usuários.

Confiabilidade: Extensão em que se pode esperar que um programa execute as funções programadas com a precisão requerida.

Eficiência: A capacidade de recursos computacionais e de código requerida por um programa para executar uma função.

Integridade: Extensão em que o acesso ao software ou aos dados por pessoa não autorizada pode ser controlado.

Usabilidade: Esforço requerido para aprender, operar, preparar as entradas e interpretar os resultados de um programa.

Manutenibilidade: Esforço requerido para localizar e corrigir um erro no programa.

Testabilidade: Esforço requerido para testar um programa a fim de garantir que ele execute suas funções estabelecidas.

Flexibilidade: Esforço requerido para modificar um programa.

Reusabilidade: Extensão em que um programa pode ser usado em outra aplicação.

Interoperabilidade: Esforço requerido para juntar um sistema com outro.

Portabilidade: Facilidade do software de operar em vários ambientes.

2.5 Planejamento do teste

2.5.1 Visão Geral

Para fazermos um bom planejamento, é necessário usar um documento padronizado, pelo menos no ambiente da empresa, mas que seja guiado por regras e normas nacionais ou internacionais. O Documento que permite fazer o planejamento de testes é conhecido como plano de testes, e é nele que definimos o nível de cobertura a abordagem dos testes. O Plano de testes deve ser elaborado usando como base os requisitos da aplicação e os requisitos de teste.

2.5.2 Importância do Plano de testes

É através deste documento que terá uma visão do que deverá ser desenvolvido, deverá constar os níveis de cobertura de testes, os envolvidos, prazos e riscos.

O Plano de Teste descreve como o teste deverá ser executado e traça uma linha mestra a ser seguida. Esse documento pode ser o Plano Global do Projeto ou um outro documento que faça parte desse primeiro

2.5.3 Padrões de Plano de Testes

Existem alguns padrões de plano de testes como PMI, QAI, CMMI, iremos abordar o IEEE 829-1998 que se refere o modelo que estamos utilizando para elaboração do plano de teste.

Elementos que devem constar do Plano – segundo o IEEE 829-1998

• identificação do Plano de Teste
• referências
• introdução
• funcionalidades a serem testadas
• riscos do processo de teste
• funções a serem testadas do ponto de vista do usuário
• funções que não serão testadas do ponto de vista do usuário
• abordagem (estratégia) dos testes
• critérios de conclusão dos testes
• critérios para interrupção e retomada dos testes;
• entregas (Plano de Teste, Casos de Teste etc.)
• ambiente de teste
• pessoal (equipe, treinamento, local etc.)
• responsabilidades
• cronograma
• plano de riscos e contingências
• aprovação dos testes
• métricas (não estão na lista do IEEE 829-1998)

2.5.4 Etapas para elaboração do Plano de testes

A montagem de uma equipe de testes pode ser feita de duas formas:

- Testar usando os próprios desenvolvedores: Tendência na informalidade nos testes haja visto que os desenvolvedores terá que conciliar as atividades de codificação e testes e por não ser especialista em testes acaba executando testes incompletos.

Criar uma equipe de especialistas em testes

- Tem um custo inicial maior, o tempo de liberação do software torna-se maior devido a execução de todos os casos de testes elaborados, mas a qualidade aumenta por se tratar de pessoas qualificadas e preparadas para executar as atividades referentes aos testes.

2.5.5 Construir o Plano de Testes

Abaixo estão os itens que compõem um bom plano de testes:

Estabelecer os objetivos dos testes: Os objetivos do teste devem ser desenvolvidos pensando na estratégia de condução dos testes, ou seja, são objetivos estratégicos, que depois deverão ser priorizados. Além disso, é necessário que esses objetivos possam ser mensurados.

Desenvolver os roteiros de testes: Os roteiros de Teste mostram muito bem a lista de casos de teste e a maneira como eles se relacionam. A fonte para a definição desse relacionamento são os casos de uso, as regras de negocio, ou até mesmo uma lista de funcionalidades.

Definir a administração do teste: O componente administrativo do Plano de Teste identifica o cronograma, os pontos de controle e os recursos necessários. Antes de elaborar o Plano de Teste, a equipe de teste já deve estar formada, pois sua primeira atividade será elaborar esse documento.

2.5.6 Escrever o plano de testes

Informações gerais

Plano

- Descrição do software que será testado.
- Equipe de teste
- Pontos de controle para o acompanhamento do projeto de teste, inclusive a identificação de locais e datas.
- Orçamento reservado para o projeto de teste.
- Identificação das organizações que participarão do teste e uma lista de etapas para o teste do software: (Cronograma e orçamento, datas e eventos de testes detalhados, e equipamentos que serão alocados, softwares, pessoas envolvidas, ferramentas e documentos).

Especificação

- Funcionalidades de negócio
- Especificação dos testes unitários, teste de integração, teste de sistema e teste de aceitação;

Avaliação

- Definição das técnicas dos critérios utilizados para avaliar os resultados dos testes

Descrições dos testes.

- Identificar os testes a serem executados;
- Identificar o método utilizado manual ou automatizado
- Entrada: Descrever os dados de entrada
- Saída: Descrever as saídas esperadas como resultado dos testes

2.6 Gerencia de Configuração

A gerencia de configuração é um conjunto de atividades para gerir modificações, identificando os produtos de trabalho que podem ser modificados, estabelecendo relacionamento entre eles, definindo mecanismos para administrar as diferentes versões desses produtos de trabalho, controlando as modificações impostas e fazendo auditoria, e preparando relatórios sobre as modificações efetuadas. Como muitos produtos de trabalho são produzidos quando o software é construído, cada um deve ser identificado de modo único. Feito isso, mecanismos para controle de versão e de modificação podem ser estabelecidos. Para garantir que a qualidade seja mantida a medida que modificações são feitas, o processo sofre auditoria.

Os objetivos do gerente de configuração constituem em garantir que os procedimentos e políticas para criar, modificar e testar o código estão sendo seguidos, bem como tornar acessível à informação sobre o projeto. Para implementar técnicas para manter controle sobre modificações de código, o gerente de configuração introduz mecanismos a fim de fazer solicitação oficial de modificação, para avaliá-las e para autorizar as modificações.

No contexto de engenharia de software, o referencial é um marco no desenvolvimento do software. O referencial é construído pela entrega de um ou mais itens de configuração de software que tenham sido aprovados como conseqüência de uma revisão técnica formal. Quando todas as partes do modelo tiverem sido revisadas, corrigidas e depois aprovadas, o modelo de projeto torna-se um referencial.

Antes que um item de configuração de software se torne referencial, qualquer modificação pode ser feita rápida e informalmente. No entanto uma vez estabelecido o referencial passamos por uma porta que da passagem a um único sentido. Modificações podem ser feitas, mas um procedimento formal especifico deve ser aplicado para avaliar e verificar cada modificação.

2.6.1 Itens de configuração de software

Um item de configuração de software é a informação como parte do processo de engenharia de software. Em caso extremo, pode-se considerar um SCI (item de configuração de software) como sendo uma seção de uma especificação grande ou um caso de teste em uma seqüência de testes grande. Mais realisticamente um SCI é um documento, toda uma seqüência de casos de teste ou um componente de programa que tem nome, conforme pode ser mostrado na figura 3 abaixo:

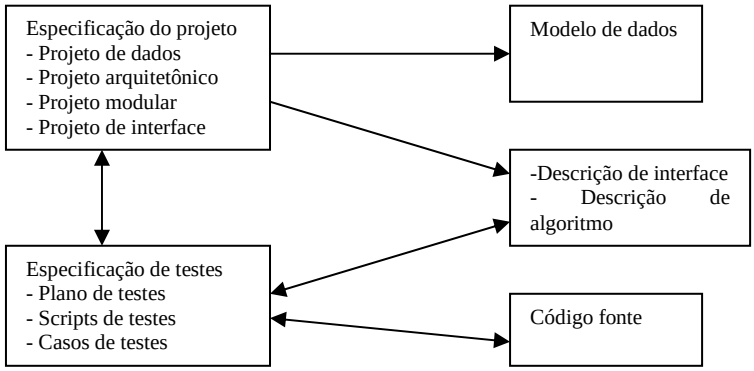

Figura 3. Objetos de configuração.

Exemplo do Plano de testes

Histórico de Revisão	Data
	Versão
	Descrição
	Autor
Introdução	Papeis e responsabilidades
	Ambiente, técnicas e ferramentas
Objetivos de testes	
Responsabilidade de testes	Testes funcionais
	Testes não funcional
Responsáveis das demais Validações e Verificações	
Validações	Validações dos requisitos com cliente
	Validações para implementação
Verificação	Revisão por pares (Requisitos e projetos
Escopo de testes	
Planejamento de testes	Casos de testes

Tabela 2. Plano de testes.

Exemplo de Plano de gerenciamento de configuração

Histórico da Revisão	Data
	Versão
	Descrição
	Autor
Identificação da configuração	Definição de itens de configuração
	Padrão de identificação
Controle da configuração	Definição das Baseline
	Atualização das Baseline
	Processo de controle de configuração
Monitoramento da situação da configuração	Armazenamento das informações
	Permissões de acesso
	Auditorias de Baseline

Tabela 3. Plano de gerenciamento de configuração.

2.7 Elaboração dos testes

A fase de elaboração dos testes tem como principal atividade de elaboração dos cenários de teste, que serão executados, testados na fase seguinte. Um cenário é que visa ajudar a solucionar um problema complexo, recriando ou visualizando um caminho a ser seguido. Um bom cenário é aquele que pode ser usado por qualquer pessoa. O cenário de teste é o caminho a ser seguido ou a situação a ser testada. O cenário que ser como base para o teste é descrito em uma especificação do sistema por exemplo é o caso de uso. O caso de teste é o cenário a ser executado para verificar se o que foi especificado está devidamente implementado.

2.7.1 Documentação do teste

A norma IEEE 829-1998 descreve um conjunto de documentos para as atividades de teste de um projeto de software. Os documentos definidos na norma cobrem as tarefas de planejamento, especificação ou elaboração e análise dos resultados.

Plano de teste: Apresenta o planejamento para a execução do teste, incluindo a abrangência, a abordagem, os recursos e a cronograma das atividades de teste. Identifica os itens e as funcionalidades a serem testados, as tarefas a serem realizadas e os riscos

associados com a atividade de teste, descrevendo ainda os diferentes ambientes que serão utilizados durante o teste.

Especificação de caso de teste: Define os casos de teste, o que inclui os dados de entrada, os resultados esperados, as ações e as condições gerais para a execução do teste.

Especificação do processo de teste: Identifica todos os passos necessários para a operação do sistema e o exercício dos casos de testes especificados, de maneira a cobrir o projeto de teste planejado. Os procedimentos de teste formam um documento esperado, que se espera que seja seguido passo a passo sem imprevistos.

Os documentos e os processos neles descritos podem ser aplicados a diversos âmbitos (comercial, cientifico) e tal aplicação não está restrita ao tamanho, a complexidade ou a severidade do software, podem ser usados para as diversas fases de teste, desde o teste de unidade até os testes de aceitação e de regressão.

É muito importante que, no início de cada projeto de software, a equipe responsável pelo teste adapte o projeto de teste, decidindo por exemplo se é mais conveniente elaborar um único plano englobe os teste de unidade, integração e aceitação, ou um plano para cada uma das fases de testes citados.

2.7.2 Elaboração do plano de caso de teste

O plano de caso de teste é o documento que registra todo o planejamento dos testes dos requisitos estabelecidos durante o ciclo de desenvolvimento do software. Este documento determina o que será testado, e seu principal objetivo consiste em identificar o maior numero de cenários e variações de determinado requisito de software. Cada cenário será representado por um conjunto de casos de testes a ser validado por uma lista de procedimentos, incorporados numa suíte de testes posteriormente executados.

2.7.3 Definição de caso de teste

Caso de teste é como a especificação mais detalhada do teste, estabelece quais informações serão empregadas durante os testes dos cenários e quais serão os resultados esperados. Para isso é necessário determinar a massa a ser utilizada no teste de modo a validar todos os requisitos do software.

O principal insumo para criação dos casos de testes são os requisitos de negócio e casos de uso.Um caso de teste é composto por um conjunto de entradas, condições de

execução, resultados esperados, tendo como objetivo verificar os requisitos especificados do sistema.

Os casos de teste estabelecem quais informações serão empregadas durante os testes dos cenários e quais serão os resultados esperados, estabelecendo a massa crítica de teste necessária para validar todos os requisitos do software.

Um bom caso de teste deve ter os seguinte padrão de qualidade

Efetivo	Testar o que se planejou testar
Econômico	Sem passos desnecessários
Reutilizável	Possa ser repetido
Rastreável	Possa identificar o requisito testado
Auto-Explicativo	Possa ser executado por qualquer pessoa

Tabela 4. Padrão de qualidade.

Exemplo de uma estrutura de casos de teste - Template

Identificação	Caso de teste CT01 – XXX
Pré-Condições	
Pos condições	
Entrada esperada	
Saída esperada	
Critérios especiais	
Sistema operacional	
Implementação	
Release	
Interdependência (Rastreablidade)	

Tabela 5. Casos de testes.

2.7.4 Fase de elaboração dos cenários do teste

Os critérios de teste servem para orientar o testador na geração dos casos de teste. Os elementos requeridos de um critério de teste são os elementos ou as características do software que deverão ser exercitados quando o software for testado. Os casos de testes refletem os requisitos que serão verificados durante os testes do software. Um requisito que não seja verificável não terá um caso de teste associado a ele. Os casos de testes podem ser classificados em varias categorias. A mais comum e importante baseia-se no negócio do software, isto é naquilo que ele precisa executar.

No mercado comercial super competitivo, não é possível testar exaustivamente todas as combinações de casos de teste de um sistema. Para contornar esse problema, existem métodos que auxiliam na escolha e critérios para a geração dos cenários de testes.

Método Step-by-step: Tem como objetivo principal produzir rapidamente casos de testes completos para especificação do sistema.

Método PairWise de teste: Ou teste de combinação dupla é um critério de teste baseado na especificação, que exige que, para cada par de parâmetro de entrada de um sistema, cada combinação de valores validos entre esses dois parâmetros seja coberta pelo menos por um caso de teste.

Método de classe de equivalência: Usada para reduzir o número de casos de teste a um nível controlável, mas mantendo uma cobertura razoável de teste. As classes de equivalência podem ser determinadas para os inputs do sistema, sendo então denominadas classes de entrada. Da mesma maneira, as saídas do sistema podem ser organizadas em classes de equivalência denominadas classes de saída.

Método dos valores limite: os casos de testes selecionados são os valores das fronteiras. O método de analise do valor-limite pode complementar o método de classes de equivalência, uma vez que a seleção de casos de teste é feita a partir das extremidades das classes de equivalência.

2.8 Execução dos testes

O Plano de Teste é fundamental para definir procedimentos necessários para implementar um processo ordenado e controlado de:

• Execução dos testes

• Registro de defeitos

• Coordenação de re-trabalho

• Controle de versão

Concluído o Plano de Teste, as atividades de preparação e execução dos testes se iniciam:

• Instalação do ambiente de teste
• Identificação dos ciclos de teste necessário para
• Replicar os processos de produção
• Relacionar os ciclos de teste com os casos de teste
• Revisar resultados
• Registrar os defeitos encontrados
• Migrar versões corrigidas para o ambiente de teste e retestá-las

Para cada fase, os testes planejados são executados e os resultados obtidos são comparados com os resultados esperados

O plano de testes deve definir também regras e responsabilidades para execução dos testes. A definição dos de quem irá executar os testes dependerá do nível dos testes executados.

Matriz de responsabilidade – Execução dos testes	
Nível	Responsáveis
Teste unitário	Desenvolvedores
Teste de integração	Desenvolvedores/ Arquitetos
Teste de sistema	Analista de testes / testador
Teste de aceitação	Usuários

Tabela 6. Matriz de responsabilidade.

As responsabilidades de cada um na execução dos testes devem ser documentadas no plano de teste. Por exemplo, os programadores são os responsáveis pela execução dos testes unitários, ao passo que os testadores são os responsáveis pela execução dos testes de sistema.

Em cada uma das etapas do processo de teste é preciso executar os testes e analisar os resultados esperados. Todos os registros da execução dos testes devem ser guardados sob ferramentas de gestão de testes, o que permite o acompanhamento do progresso dessa execução.

2.8.1 Teste Unitário

Os testes unitários serão feitos pelos programadores como forma de garantir o funcionamento adequado do programa. Serão feitos testes unitários das principais rotinas ou das rotinas mais criticas do projeto.

2.8.2 Teste de Integração

O teste de integração terá inicio quando os componentes a ser integrados já tiverem sido passados pelo teste unitário. O teste de integração tem como objetivo garantir que os componentes da aplicação, ou daquele módulo da aplicação, possam ser integrados com sucesso em uma determinada funcionalidade.

2.8.3 Teste de Sistemas

O teste de sistema precisa garantir que os requisitos do software foram cumpridos e implementados corretamente. O teste de sistema deverá ter inicio quando o teste de integração estiver sido encerrado. Para que o sistema seja executado com sucesso é necessário seguir alguns procedimentos.

- Ambiente de testes próximo do ambiente de produção;
- Definir quais casos de testes será utilizado na execução do sistema;
- Ferramenta para registrar bugs.

2.8.4 Teste de Aceitação

O teste de aceitação será feito pelo analista de testes ou analista de sistemas com os usuários para garantir que tudo que foi definido por eles nos requisitos tenha sido incluído no produto que esta sendo entregue.

2.8.5 Quando Parar de Testar

O numero de defeitos encontrados que não foram corrigidos, considerando o grau de severidade que classificamos como: Baixo, médio e alta prioridade (Bugs críticos, impeditivos).

Outro método utilizado é avaliando os riscos com os riscos da não liberação, diante da possibilidade de perder negócio com cliente, muitas vezes vale a pena correr o risco de liberar a aplicação para produção mesmo que seja preciso tratar um defeito encontrado algum tempo depois já em produção.

2.8.6 Gestão de defeitos

Os defeitos são resultados de erros existentes no software ou em outros artefatos desenvolvidos pelas equipe envolvida no processo de desenvolvimento do software. A gestão de defeitos é realizada da através de alguns processos como:

Prevenção de defeitos
Baseline a ser entrege
Identificação do defeito
Solução do defeito
Melhoria do processo
Relatórios de gestão

2.8.7 Prevenção de Defeitos

Cada vez mais as novas técnicas de teste de software buscam diminuir o número de defeitos. Isto equivale a dizer que os analistas de testes têm esforçado mais e mais para cooperar com a equipe de desenvolvimento na verificação da documentação do software. Já na fase de levantamentos de requisitos a verificação em pares por outro analista de sistemas tem como objetivo de encontrar erros primários, pois a maneira mais elementar de prevenir defeitos e detecta-los nos estágios iniciais do desenvolvimento do software. Outra verificação realizada é a detecção dos riscos do projeto de desenvolvimento e teste e tentar mitiga-los. Alguns riscos que são analisados:

Falta de requisitos importantes
Inconsistência dos requisitos
Falta de conhecimento pela equipe de testes/ desenvolvimento com as regras de negócio
Falta de conhecimento por algum hardware ou software envolvido no projeto
Equipe mal qualificada
Rotatividade da equipe envolvida no projeto

2.8.8 Baseline a ser Entregue

Um produto a ser entregue e considerado baseline quando atinge o marco pré-definido no seu desenvolvimento. Depois que o produto é entregue ao cliente, o sistema fica disponível para o gerenciamento da configuração. Um defeito é caracterizado quando um componente de um produto (baseline) não satisfaz a seu conjunto de requisitos.

2.8.9 Identificação de Defeitos

A identificação dos defeitos é realizada através dos testes e revisões em pares ou pelos usuários na utilização do sistema já em produção. Após identificar os defeitos é feito o registro na ferramenta de registro de bugs chamada Titan. Essa ferramenta é um bugtracker que foi desenvolvido internamente, com objetivo de introduzir os bugs que encontramos numa aplicação e onde podemos depois acompanhar e controlar a evolução de cada bug.

A devida classificação de acordo é de acordo com o impacto, tanto para o sistema quanto para os negócios. A classificação é realizada da seguinte forma:

Baixa: Não afeta a utilização do sistema como erro na documentação ou interface gráfica

Media: Defeito na tela mas não impede o uso do sistema.

Alta: Produz resultado errado ou sistema para de funcionar.

As técnicas utilizadas são as seguintes:

Técnica estática

O produto é examinado com objetivo de encontrar defeitos, Executada no inicio do projeto são muitos eficientes. Exemplo utilizado é a revisão por pares.

Técnica dinâmica

O produto é usado com objetivo de encontrar defeitos, exemplo a realização dos testes funcionais do sistema.

2.8.10 Solução do Defeito

Depois de registrado os bugs pelo equipe de desenvolvimento ou usuário final, o gerente do projeto programa a correção de acordo com a prioridade e recursos disponíveis.

2.8.11 Relatório de Gestão

É realizado um registro completo dos desvios identificados durante o processo de testes para que sirva de base no decorrer do projeto, no que diz respeito a medições de qualidade. Os relatórios de gestão são gerados pela ferramenta desenvolvida internamente.

2.8.12 Teste de Aceitação

O teste de aceitação é realizado para garantir que o software foi desenvolvido ou a parte que esta sendo entregue para testes esta correto, e que já pode ser usado em produção

pelos usuário final. O teste de aceitação é feito com o usuário final no ambiente real utilizando a versão beta. A preocupação do usuário final e da equipe de testes que realiza o teste de aceitação com o cliente e a de garantir que o software esteja fazendo exatamente aquilo que foi pedido na fase de levantamento de requisitos.

O teste de aceitação normalmente conduzido pelos usuários, não deve necessariamente ser executado apenas no fim do processo de desenvolvimento, mas durante todas as suas etapas, quando produtos intermediários podem ser testados **conforme figura 3.**

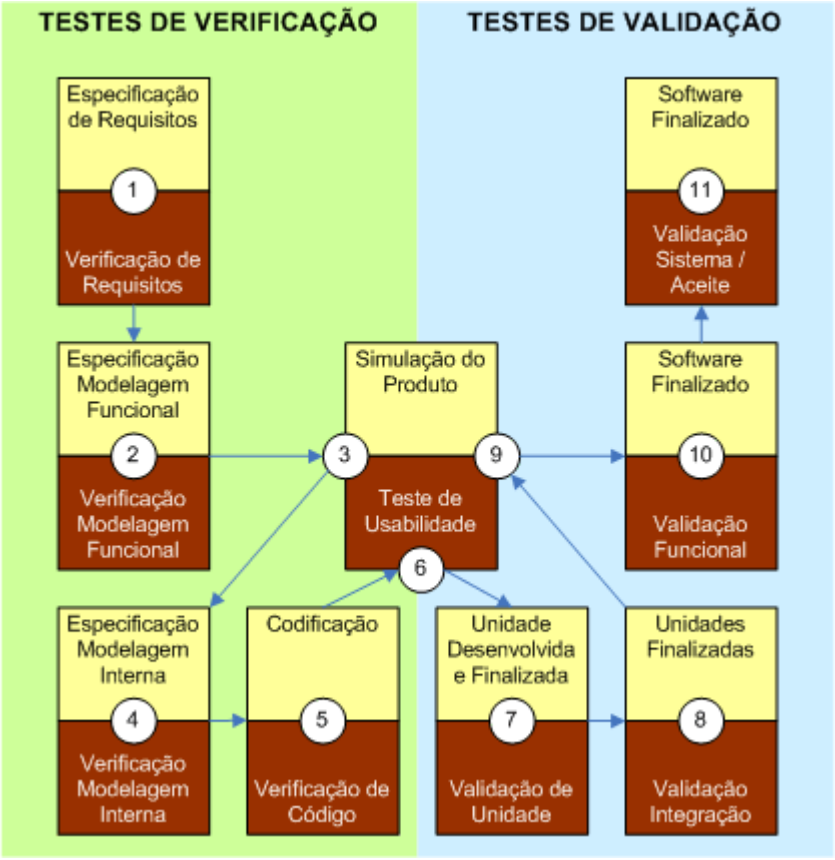

Figura 3. Testes de verificação e validação
Disponível em < http://www.rsinet.com.br/?q=node/18 > Acesso em 08 Dez. 2010

2.8.12.1 Plano de Aceitação

Para obter o aceite do cliente o plano de aceitação deve conter as seguintes informações:

Descrição do projeto:
Possui uma descrição detalhada do projeto

Responsabilidade dos envolvidos:
Possui as responsabilidades e nomes de cada envolvido durante o teste de aceitação

Requisitos cobertos:
Lista de todos requisitos e casos de uso cobertos no teste de aceitação

Assinatura do Cliente:

Assinatura do cliente ciente que foi cumprido tudo que foi combinado no inicio do projeto.

2.8.12.2 Execução do Teste de Aceitação

A execução do teste de aceitação segue os critérios definidos no plano de aceitação. Nesta etapa esse teste será a ultima oportunidade do usuário verificar se o produto que foi desenvolvido esta de acordo com suas necessidades. O usuário também deverá verificar toda a documentação do software que esta sendo entregue, se esta de acordo com o que foi desenvolvido foi de fato o que foi solicitado.

3 Aplicando o modelo de verificação e Validação no desenvolvimento dos softwares

3.1 O modelo em V

O modelo em V é uma variação do modelo cascata, que demonstra como as atividades de teste estão relacionadas com a análise e o projeto. O modelo em V sugere que os testes de unidade e de integração também podem ser utilizados para verificar o projeto do programa. Isto é durante os testes de unidade e de integração, os programadores e a equipe de testes deveriam garantir que todos os aspectos do projeto do sistema foram implementados corretamente no código. De maneira semelhante, os testes de sistema, deveriam verificar o projeto do sistema, assegurando que todos aspectos do sistema foram corretamente implementados. O teste de aceitação, conduzido mais pelo cliente do que pelo desenvolvedor, valida os requisitos associando uma etapa de teste com cada elemento da especificação. Esse tipo de teste confirma que todos os requisitos foram corretamente implementados antes de o sistema ser aceito e pago.

A conexão entre os lados esquerdo e direito em V conforme figura 4, implica que, caso sejam encontrados problemas durante a verificação e a validação, o lado esquerdo do V pode ser executado novamente para corrigir e melhorar os requisitos, o projeto e a codificação, antes da execução das etapas de teste do lado direito do V. Em outras palavras, o modelo em V torna mais explicitas algumas interações e repetições do trabalho, ocultas no modelo cascata. Enquanto o modelo em cascata está nos documentos e nos artefatos, o enfoque do modelo em V esta na atividade e na correção.

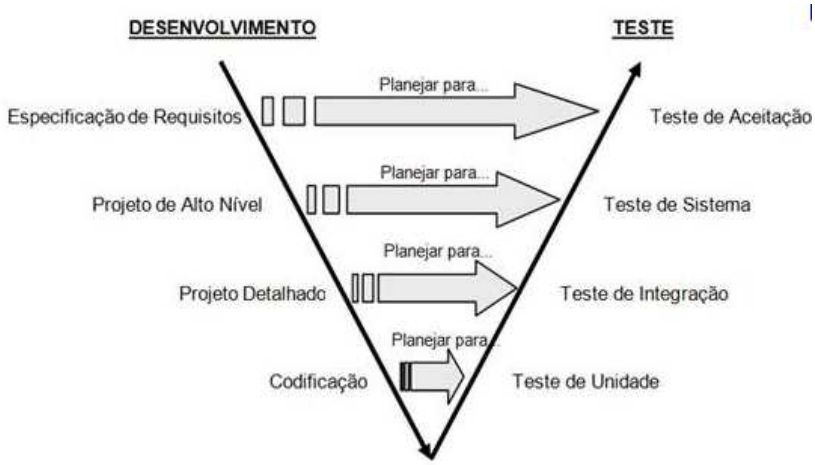

Figura 4. Modelo em V

Disponível em < **http://www.devmedia.com.br/articles/viewcomp.asp?comp=8035** > Acesso em 08 Dez. 2010

3.2 Principais Participantes do Processo

***** Patrocinador – Responsável por identificar a necessidade e justificar o desenvolvimento do sistema. Conduzir o estudo das alternativas, análise de viabilidade e risco.

* Gerente de Projeto – Responsável por conduzir o projeto de desenvolvimento adequadamente, atendendo os requisitos, prazo, custo e realiza toda documentação.

* Gestor de Tecnologia da Informação – Responsável pelas atividades de TI (instalações, gestão de bases de dados, suporte técnico, help-desk, definição das políticas e padrões.)

* Gestor de Qualidade – Responsável pelas ações de controle de qualidade dos produtos, certificando que foi desenvolvido de acordo com as especificações e atendendo as expectativas do usuário.

* Arquiteto de Software – Responsável por elaborar a arquitetura do software e garantir a implementação de acordo com a arquitetura.

* Analista de sistemas – Responsável por levantar requisitos, apoio às atividades de planejamento e modelagem, especificação, acompanhamento da construção dos testes, testes de integração, documentação.

* Programador – Elaboração e execução dos testes unitários do código, teste de integração, e codificação.

* Analista de Testes – Modelagem e elaboração dos casos de testes manuais e automatizados.

* Testador – Execução dos casos de testes e scripts.

3.3 Implementação na empresa

O modelo de verificação e validação esta sendo implementado na empresa Eteg que desenvolve soluções especificas para sua empresa e ainda fornece consultoria e treinamento proporcionando total autonomia na gestão, tendo como objetivo atingir o nível C do mps.Br.

4 Validação

De acordo com mps.Br o propósito de VAL é confirmar que um produto ou componente do produto atenderá a seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido. O cenário atual da empresa foi elaborado pelo gerente de projetos, que ficou responsável por avaliar cada um dos sete itens que fazem parte da validação.

4.1 Resultados esperados da validação x cenário atual

Resultados Esperados VALIDAÇÃO	Cenário Atual VAL na empresa
VAL1. Produtos de trabalho a serem validados são identificados	Não existia identificação de produtos de trabalho a serem validados
VAL2. Uma estratégia de validação é desenvolvida e implementada, estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação.	A validação é feita através de protótipos de telas. Existem testes de aceitação realizados entre o analista de testes e o cliente. Testes unitários não são realizados pelos desenvolvedores.
VAL3. Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido	Não existe critério formalizado de validação, sem ambiente de validação.
VAL4. Atividades de validação são executadas para garantir que os produtos de software estejam prontos para uso no ambiente operacional pretendido.	A validação é feita junto ao cliente de acordo com o cronograma do projeto. Alguns projetos possuem casos de teste e plano de testes. Os casos de testes são executados de acordo conforme o plano de testes.
VAL5. Problemas são identificados e	Os bugs ou qualquer inconformidade no

registrados.	projeto são registrados em uma ferramenta desenvolvida internamente de registro de bugs chamada Titan.
VAL6. Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas.	O gerente avalia os resultados através dos relatórios colhidos na ferramenta desenvolvida internamente (Titan) para gerenciar projetos.
VAL7. Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas.	As evidencias de que os produtos foram desenvolvidos fica registrado na tarefa do desenvolvedor que altera o status da atividade para realizado na ferramenta chamada Titan.

Tabela 7. Cenário atual de validação na empresa.

5 Verificação

O propósito de VER é confirmar que cada serviço e ou produto de trabalho do processo ou do projeto atende apropriadamente os requisitos especificados. O cenário atual da empresa foi elaborado pelo gerente de projetos, que ficou responsável por avaliar cada um dos seis itens que fazem parte da verificação.

5.1 Cenário atual do processo de verificação

Resultados Esperados VERIFICAÇÃO	Cenário Atual VER na empresa
VER1. Produtos de trabalho a serem verificados são identificados.	Não existe identificação de produtos de trabalho para serem verificados
VER2. Uma estratégia de verificação é desenvolvida e implementada, estabelecendo cronograma, revisores envolvidos, métodos para verificação e qualquer material a ser utilizado na verificação.	É realizados testes funcionais pelo analista de teste em alguns projetos.
VER3. Critérios e procedimentos para verificação dos produtos de trabalho a serem verificados são identificados e um ambiente para verificação é estabelecido.	Não existe critério formalizado de verificação.

VER4. Atividades de verificação, incluindo testes e revisões por pares, são executadas.	Não é realizado nenhuma revisão por pares.
VER5. Defeitos são identificados e registrados.	Os defeitos encontrados são registrados em uma ferramenta desenvolvida internamente chamada Titan.
VER6. Resultados de atividades de verificação são analisados e disponibilizados para as partes interessadas.	Não existe analise de resultados de verificação.

Tabela 8. Cenário atual de verificação na empresa.

6 Mapa de Testes de Validação e Verificação

Muitos fatores podem ser identificados, mas a maioria deles tem uma única origem: erro humano. Como a maioria das atividades de engenharia, a construção de software depende principalmente da habilidade, da interpretação e da execução das pessoas que o constroem; por isso erros acabam surgindo, mesmo com a utilização de métodos e ferramentas de engenharia de software.

Para que tais erros não perdurem, ou seja, para serem descobertos antes de o software ser liberado para utilização, existe uma série de atividades chamadas de validação e verificação, com a finalidade de garantir que tanto o modo pelo qual o software esta sendo construído quando o produto em si estejam em conformidade com o esperado. Atividades de verificação e validação não se restringem ao produto final. Ao contrário, podem e devem ser conduzidos durante todo processo de desenvolvimento do software, desde a sua concepção, e engloba diferentes técnicas. As atividades de verificação são estáticas e dinâmicas. As estáticas são aquelas que não requerem a execução ou mesmo a existência de um programa ou modelo executável para para serem conduzidas (qualidade interna). As dinâmicas são aquelas que se baseiam na execução de um programa ou de um modelo (qualidade externa). Vide figura abaixo:

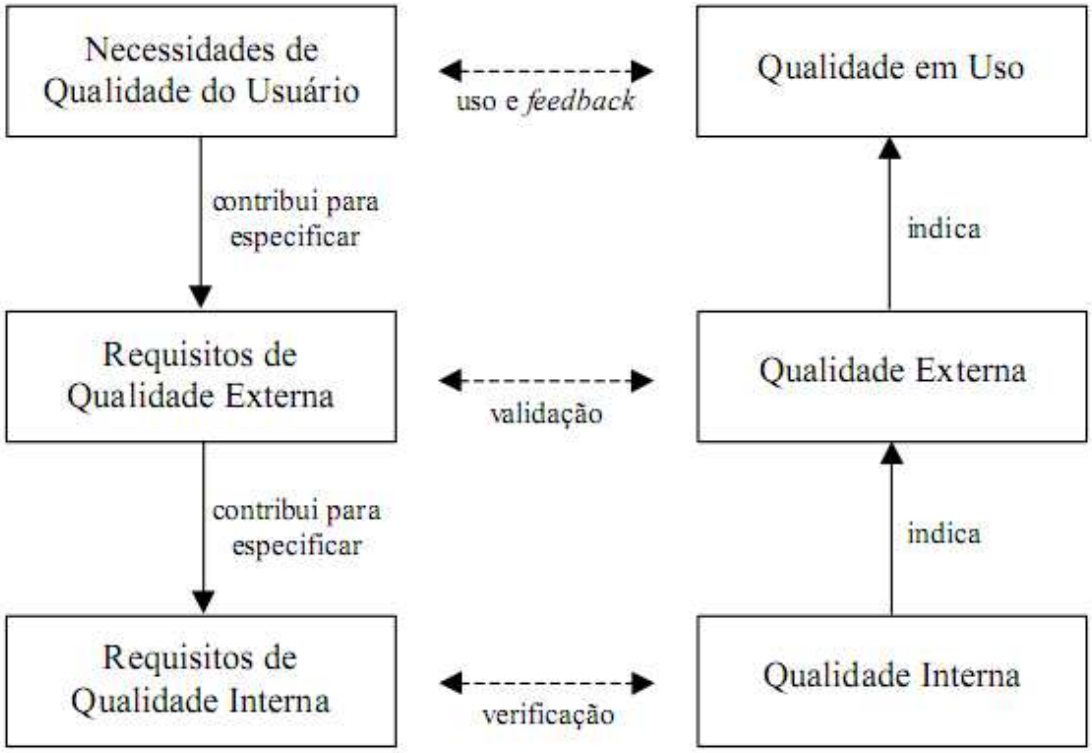

Figura 5: Qualidade interna e externa
Fonte: Próprio autor

·**Necessidades de qualidade do usuário**

A execução de um produto que satisfaça as necessidades do usuário normalmente requer uma abordagem iterativa no desenvolvimento do software, com contínuo feedback da perspectiva do usuário.

·**Requisitos de qualidade externa**

É a visão externa que inclui requisitos derivados das necessidades de qualidade do usuário, incluindo requisitos de qualidade em uso. Requisitos de qualidade externa são utilizados como alvo para validação em vários estágios de desenvolvimento.

· **Requisitos de qualidade interna**

Requisitos de qualidade interna podem ser usados como alvos para validação em vários etapas de desenvolvimento. Requisitos de qualidade interna são usados para especificar propriedades de produtos intermediários. Eles podem incluir modelos estáticos e dinâmicos, outros documentos e código fonte.

· Qualidade interna

É a totalidade de características do produto de software na visão interna. A qualidade do produto de software podem ser melhoradas durante a implementação, revisão e teste unitário.

· Qualidade externa

É a totalidade de características do produto de software do ponto de vista externo. Os testes são executados no ambiente similar ao ambiente de produção, durante a execução dos testes a maioria dos bugs são encontrados e eliminados.

· Qualidade em uso

É a visão do usuário da qualidade do produto de software quando é utilizado em ambiente de produção.

Mapa da qualidade de testes – Verificação e Validação

A seguir a tabela 9 lista os tipos de teste aplicado no desenvolvimento do software desde a verificação sendo executado o teste unitário pelo desenvolvedor até a validação onde é feito o teste de aceitação pelo cliente

Tipos de Testes - Unidade				
Artefato/Produto	Procedimento	Método	Critério	Revisor
Código-Fonte	Teste unitário	VER	Manuais ou automatizados	Desenvolvedor
Tipo de Testes - Integração				
Artefato/Produto	Procedimento	Método	Critério	Revisor
Interface,	Teste de integração	VER	checklist	Analista de sistemas/Arquiteto
Modelo de dados	Teste de integração	VER	Revisão por par	Arquiteto
Diagrama de	Teste de	VER	Revisão por par	Arquiteto

componentes	integração			
Diagrama de pacotes	Teste de integração	VER	Revisão por par	Arquiteto
Código-fonte	Caixa Branca - Integração entre as classes/funções	VER	Codificando	Desenvolvedor
script de testes	Teste de regressão	VER	Manuais ou automatizados	Analista de testes
Tipo de Testes - Sistema				
Artefato/Produto	Procedimento	Método	Critério	Revisor
Casos de uso	Teste de analise	VER	checklist/Revisão por par	Analista de sistemas
Requisitos funcionais e não funcionais	Teste de requisitos	VER	Revisão por par	Analista de sistemas/Testes
Diagrama de classes Diagrama de atividades	Teste de analise Teste de analise	VER	checklist/Revisão por par checklist/Revisão por par	Analista de sistemas
Diagrama de estado	Teste de analise	VER	checklist/Revisão por par	Analista de sistemas
Plano de testes	testes	VER	checklist/Revisão por par	Analista de testes
Projeto de testes	testes	VER	checklist/Revisão por par	Analista de testes
Versão para	Teste de	VER	Manuais ou	Analista de testes

testes -Casos de testes	Funcionalidade		automatizados	
script de testes	Testes de permance	VAL	Manuais ou automatizados	Analista de testes
Casos de testes	Testes de permance	VAL	Manuais ou automatizados	Analista de testes
Tipo de testes - Aceitação				
Artefato/Produto	Procedimento	Método	Critério	Revisor
Requisitos	Homologação	VER	Manuais	Analista de sistemas/Testes/Cliente
sistema	Aceite sistema	VAL	Manuais	cliente

Tabela 9. Mapa de qualidade.
Fonte: Próprio autor

7 Contexto de Melhoria

O processo de verificação e validação é realizado pela diretoria de projetos em conjunto com a equipe de engenharia de software que tem os seguintes objetivos:

Estabelecimento de um mecanismo padrão e otimizado para o desenvolvimento de aplicações para empresa Melhorar a qualidade dos produtos e serviços prestados aos clientes.

Garantir certificação no modelo MPS.br.

Este processo se aplicará a todos os projetos de software dentro do contexto da empresa e deverá ser seguido por seus participantes, de acordo com os papéis especificados, independentemente da tecnologia adotada.

As principais características adotadas são baseadas nos modelos:

UML, RUP, PRAXYS e MPS.BR

Incorporação de características particulares advindas de experiências de desenvolvimento de sistemas anteriores.

7.1 Proposta de Melhoria

Processos	Melhorias propostas
VAL1 / VER1	- Para estes resultados definimos a necessidade de inclusão, no Plano de Testes dos projetos, da relação dos produtos de trabalho que deverão ser validados e/ou verificados. O Plano de Testes, junto com as diretrizes para seleção dos produtos de trabalho (a serem definidas pela equipe de engenharia de software) são as principais evidências para o atendimento dos resultados. - Criação de template do plano de testes
VAL1	Para o atendimento deste resultado definimos um conjunto padrão de testes e validações, que deveriam ser executados nos projetos e previstos no planejamento dos mesmos: - Validação de requisitos e obtenção de aceite: validação dos requisitos com os usuários/clientes que os definiram e formalização da obtenção do aceite dos requisitos com os mesmos. - Testes de unidade: testes funcionais do tipo caixa-branca, realizados pelo desenvolvedor. - Testes de sistemas: testes funcionais do tipo caixa-preta, realizados pelo testador. - Testes não funcionais: testes de características como usabilidade, performance, etc.

	- Testes de aceitação e validação final: testes realizados junto com o cliente na homologação do produto final ou de um caso de uso específico, devendo ter o aceite formalizado. - Todos estes tipos de validações já foram implantados nos projetos
VAL 3 / VER 3	O Plano de Testes também deve fazer menção ao ambiente onde as validações e verificações serão realizadas. A definição detalhada dos ambientes de trabalho deve ser incluída no Plano de Gerenciamento de Configuração
VAL 4/ VER 4	O atendimento destes resultados implica na demonstração de que as atividades de validação e verificação foram, efetivamente executadas. Para tal, deverão existir dois tipos de evidências: Evidências de planejamento: - Atividades previstas e concluídas do MS-Project. - Tarefas executadas no Titan (ferramenta interna de gerencia de projetos) com as respectivas horas alocadas - Além do Plano de Testes Evidências produzidas pelas validações e verificações: - Especificações de Casos de Teste - Planilha de Registro de Verificação e Validação.
VAL 5 / VER 5	As principais evidências para a comprovação do atendimento destes resultados são as issues de bugs, problemas, melhorias, criadas no Titan a

	partir das validações e verificações realizadas
VAL 6 / VER 6	O atendimento destes resultados implica no planejamento da forma como a comunicação será feita e na realização da comunicação propriamente dita. O Plano de Gerenciamento de Comunicação deve conter todas as definições sobre as informações que serão geradas, e fornecidas, aos envolvidos (stakeholders) dos projetos, correspondendo a uma importante evidência para estes resultados. - Uma forma de comunicação prevista nos projetos da empresa - Criado uma Planilha de Registro de Verificação e Validação, onde a mesma é anexada ao e-mail.
VAL 7	A principal evidência deste resultado é o Termo de Aceite que formaliza a aceitação do produto por parte do cliente. As evidências de VAL 6, como a comunicação dos resultados e a Planilha de Registro de Verificação e Validação, também complementam a demonstração de que o resultado foi obtido, na medida em que comprovam que o produto está pronto para uso.

Tabela 10. Proposta de melhoria.

Fonte: Próprio autor

8 Estudo de Caso - Implantação do dos processos VAL e VER

O objetivo desta tabela é listar o estudo de caso realizado em um cliente chamado FHEMIG. O estudo esta sendo feito pela equipe de processo de software denominada SEPG, que é responsável por avaliar todos os processos de verificação e validação bem como propor soluções para as observações levantadas junto ao cliente.

Resultado de Processo	**Implantação Projeto Fhemig**	**Apoio SEPG**	**Pendência Organizacional / SEPG**	**Observação**
VAL				
VAL 1	Parcialmente Implantado	Sim	Sim	Definição de diretrizes para validação dos produtos de trabalho (SEPG) e formalização dos produtos de trabalho para validação no Plano de Testes.
VAL 2	Parcialmente Implantado	Sim	Sim	Formalização das validações de requisitos com o cliente e refinamento dos testes não funcionais.
VAL 3	Parcialmente Implantado	Sim	Sim	Definição de critérios e procedimentos para validação dos produtos de trabalho (SEPG) e inclusão de definição do ambiente no Plano de Testes e no Plano de Gerenciamento de Configuração.
VAL 4	Parcialmente Implantado	Não	Não	Utilização da

				Especificação de Caso de Teste e efetivação dos registros de validação.
VAL 5	Parcialmente Implantado	Não	Não	Efetivação dos registros de validação.
VAL 6	Parcialmente Implantado	Não	Não	Falta elaborar o Plano de Gerenciamento de Comunicação e definir a forma de comunicação dos resultados.
VAL 7	Parcialmente Implantado	Não	Não	Formalização das homologações com o cliente.
VER				
VER 1	Parcialmente Implantado	Sim	Sim	Definição de diretrizes para verificação dos produtos de trabalho (SEPG) e formalização dos produtos de trabalho para verificação no Plano de Testes.
VER 2	Parcialmente Implantado	Sim	Sim	Repasse de critérios para orientação da verificação (revisão por pares) nos artefatos do projeto.
VER 3	Parcialmente Implantado	Sim	Sim	Definição de critérios e procedimentos para verificação dos produtos de trabalho (SEPG) e inclusão de definição do ambiente no Plano de Testes e no Plano de

				Gerenciamento de Configuração.
VER 4	Parcialmente Implantado	Não	Não	Utilização da Especificação de Caso de Teste e efetivação dos registros de verificação.
VER 5	Parcialmente Implantado	Não	Não	Efetivação dos registros de verificação.
VER 6	Parcialmente Implantado	Não	Não	Falta elaborar o Plano de Gerenciamento de Comunicação e definir a forma de comunicação dos resultados.

Tabela 11. Implantação do processo de validação e verificação.
Fonte: Próprio autor

Estas informações devem ser utilizadas na atualização do plano de ação do SEPG. A coluna "Apoio SEPG" informa onde o apoio direto do SEPG, para a implantação das práticas relacionadas aos respectivos resultados, é prioritário. No caso de VAL e VER, o SEPG deverá atuar na definição de diretrizes para validação e verificação dos produtos de trabalho (VAL 1 e VER 1), na definição de critérios e procedimentos para validação e verificação dos produtos de trabalho (VAL 3 e VER 3), na definição de critérios para testes não funcionais (VAL 2) e na definição de critérios para orientação da verificação (revisão por pares) nos artefatos do projeto (VER 2).

A coluna "Pendência Organizacional / SEPG" indica se, a partir da implantação das práticas nos projeto, identificamos alguma questão que deve ser tratada em um nível mais alto, compondo o processo organizacional. Os critérios envolvidos em VAL 1, VAL 2, VAL 3, VER 1, VER 2 e VER 3 devem compor os processo organizacionais, não sendo específicos dos projetos da Fhemig.

Para o atendimento do VAL 2 é necessário efetivar a formalização das validações de requisitos com o cliente, implantada inicialmente no projeto Fhemig.

O Plano de Testes atenderá, os processos VAL 1, VAL 3, VER 1 e VER 3.

Para VAL 4, VAL 5, VER 4, e VER 5 é necessária a efetivação do registro dos resultados das verificações e validações. Já existe uma proposta de uma Planilha de Registro de Verificação e Validação. Para VAL 4 e VER 4 também é fundamental a utilização da Especificação de Caso de Teste, cujo template atualizado deve implantado.

O Plano de Gerenciamento de Comunicação, a ser elaborado para os projetos da Fhemig, possibilitará o atendimento de VAL 6 e VER 6, definindo as formas de comunicação relacionadas às validações e verificações realizadas. O VAL 7 necessita de uma formalização das homologações realizadas com o cliente, além do aceite fornecido pela coordenação do SIGH.

9 Conclusão

O papel desempenhado pela equipe de testes é fundamental para o sucesso do software, pois os testes servem como parâmetro para o sucesso ou não do produto criado. Em relação aos processos de VAL e VER se observa que ambos são complementares, quando analisamos as atividades de Testes. Estes processos possuem objetivos diferentes, mas colaboram para a qualidade final do software. Outro aspecto relevante neste contexto de melhoria é a alta importância dada para os testes como prática de verificar e validar o software. Avaliando os resultados esperados destes processos, outras técnicas podem ser avaliadas para implementação na empresa como cenários de casos de uso, revisão por pares, testes unitários e funcionais além de melhorias na prática atual de prototipação.

Observa-se também que o comprometimento dos colaboradores, a percepção da importância das práticas dos processos de VER e VAL e o apoio da alta direção será fundamental para o sucesso da implementação destes processos na organização. Além disto, orienta-se que a melhoria dos processos da organização seja conduzida de forma contínua e gradativa de acordo com as suas necessidades e objetivos estratégicos, contribuindo cada vez mais para a satisfação dos clientes e qualidade dos produtos.

Para que uma empresa obtenha o nível C do mps.br o processo de testes tem que estar implantado e enraizado na cultura da empresa pois o teste de software é uma atividade dinâmica onde o testador executa o programa ou modelo utilizando algumas entradas particulares e verifica se seu comportamento esta de acordo com o esperado. Os dados de tal informação podem servir como fonte de informação para a localização e a correção dos defeitos.

Concluímos que o processo de testes encontra-se ainda na fase parcialmente implantado conforme apresentado no estudo de caso do processo de validação e verificação, mas os ganhos já são visíveis na empresa seja na satisfação do cliente que esta recebendo um produto de qualidade ou na própria equipe envolvida no projeto que produz um produto com mais qualidade interna e externa.

O processo de desenvolvimento deve estar na busca sempre pela melhoria continua dos produtos e serviços prestados. Podemos considerar que o objetivo foi atingido parcialmente com aquisição do nível F do mps.Br. O objetivo final será atingir o nível C proporcionando para o cliente e equipe de desenvolvimento, uma melhor qualidade do produto e ou serviço.

REFERÊNCIAS

INTHURN, Cândida. Qualidade e teste de software. 1ª edição, Florianópolis: Visual Books, 2001.

MALDONADO, Jose; DELAMARO, Márcio; JINO, Marcio. Introdução ao teste de software.Rio de Janeiro: Campus, 2007.

MOREIRA FILHO, Trayahú R.; RIOS, Emerson. Projeto & engenharia de software: teste de software. Rio de Janeiro: Alta Books, 2003.

MOREIRA FILHO, Trayahú R.; BASTOS, Anderson; RIOS, Emerson; CRISTALLI Ricardo. Base de conhecimento em teste de software. São Paulo: Martins,2007.

MPS.BR (2009) "Associação para Promoção da Excelência do Software Brasileiro. MPS.BR – Guia Geral", http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf

PAULA FILHO, Wilson de Pádua. Engenharia de software: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

PRESSMAN, Roger S. Engenharia de software. São Paulo: Makron Books, 1995.

PORTO, Josiane Brietzke et al. (2007). "A Experiência de Avaliação MPS.BR Nível F na Qualità", Revista ProQualiti Vol. 3, N. 3.

PETERS, James; PEDRYCZ, Witold. Engenharia de software. Rio de Janeiro: Campus,2001.

RON, Patton. Software testing. Indianápolis, Sams, 2005.