FACULDADE DE TECNOLOGIA DE SÃO PAULO

CAMILA ROCHA

Estudo da qualidade de software na Metodologia V-model e sua interação com metodologias ágeis (SCRUM).

São Paulo

2011

FACULDADE DE TECNOLOGIA DE SÃO PAULO

CAMILA ROCHA

Estudo da qualidade de software na Metodologia V-model e sua interação com metodologias ágeis (SCRUM**).**

Monografia submetida como exigência

parcial para a obtenção do Grau

de Tecnólogo em Processamento de Dados

Professor Orientador: DIONISIO GAVA JÚNIOR

São Paulo

2011

À minha família que sempre esteve ao meu lado em todos os momentos.

Ao meu futuro marido que sempre me apoiou

AGRADECIMENTOS

Ao meu orientador, professor Dionísio Gava Júnior por todos os ensinamentos e por agregar no meu desenvolvimento profissional.

Aos meus amigos da FATEC-SP por toda ajuda e colaboração em todos esses anos de curso.

RESUMO

Cada vez mais a necessidade das empresas de trazer satisfação para os seus clientes aumentou a importância de alguns fatores para as empresas. Como hoje as pessoas vivem na era do conhecimento, trazendo como principal foco o cliente, mais e mais há a necessidade da qualidade em todos os produtos e isso não seria diferente com o produto software desenvolvido por milhares de consultorias ao redor do mundo.

Com o surgimento dessa nova era foi criado uma nova área no mundo do software: o QA (*Quality Assurance*) – a área da qualidade, no qual seu time é feito de testes ou testadores cuja função é encontrar os possíveis problemas que o usuário – cliente - possa encontrar que podem ser referentes ao *developer* (desenvolvedor), no qual possa não ter implementado ou até mesmo o ambiente não conseguir comportar a aplicação fabricada. Isto é, são vários fatores que um *tester* tem que verificar, rastrear e informar para levar qualidade para o cliente.

Com isso, percebesse que o cliente nem sempre escolhe o melhor preço, hoje principalmente na área de software o primeiro tópico é a qualidade que bem gerenciada e planejada de forma rápida se torna um fator de grande vantagem no mercado competitivo. Para tudo isso, esse trabalho irá mostrar a metodologia em Cascata com o V-model e sua integração com SCRUM criando maior agilidade nos processos.

ABSTRACT

Increasingly, the need for companies to bring satisfaction to their clients increased. As we live in the knowledge era, in which the customer is the focus more and more there is the need for quality in all products and this is no different with the Software products developed by thousands of consultants around the world.

With the emergence of this new era has created a new area in the software world: the QA (quality assurance) - the area of quality, in which his team will compose of testers whose job is to find potential problems that the user - can client-find that the developer cannot be implemented or even the environment may not have behaved the application made. That is, there are several factors that a tester has to verify, track and report to bring quality to the customer.

We realize that the customer does not always choose the best price, today mainly in the software the first topic is the quality that well planned and managed quickly, in which we can cite the agile methodologies (SCRUM) becomes a factor great advantage in the competitive market.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ABNT: Associação Brasileira de Normas Técnicas.

CMM: Capability Maturity Model.

IEEE: Instituto de Engenharia Elétrica e Eletrônica.

ISO: InternationalOrganization for Standardization.

QA: Quality Assurance.

SUMÁRIO

1 INTRODUÇÃO

Com objetivo em mostrar os benefícios da qualidade nos sistemas desenvolvidos nas muitas consultorias de T.I. existentes o trabalho visa compreender melhor a área de teste pela metodologia em cascata V-Model; além de mostrar a importância que essa área pode proporcionar para as empresas.

Os benefícios que serão abordados neste trabalho irá mostrar como a qualidade é um meio atrativo para a conquista de novos clientes tanto do lado do produto (com qualidade) como do financeiro. Isto é, muitas empresas de tecnologia da informação usam argumentos que evitam a criação de uma área de testes (QA) devido ao custo adicional na criação de seus produtos, softwares. Com isso, pretendesse desmistificar essa crença e mostrar que além do benefício de um melhor produto para o cliente há também a possibilidade de maiores lucros para empresa.

Além da qualidade no produto existe a necessidade de uma qualidade nos processos. Com isso, o trabalho mostrará através do V-model (metodologia em cascata) que a qualidade dos softwares é garantida através desse modelo explicando seus processos, tipos, técnicas de testes entre outros, como também como as padronizações destes são necessárias através das ISOs entre outras normatizações.

O cenário que o “nascimento” da qualidade se mostrará será no momento em que a competitividade entre as empresas aumentou ao mesmo tempo em que os clientes ficavam mais exigentes devido a grande oferta de produto; além da maior disseminação das informações. Com isso, houve uma necessidade dos serviços serem melhores tanto na entrega quanto no produto, no qual foi preciso criar metodologias que ajudassem nesses dois fatores, qualidade e eficácia.

Para isso, esse trabalho irá mostrar além do modelo em V (qualidade) sua associação com o SCRUM (agilidade) criando eficácia na entrega dos produtos gerando maior satisfação e fidelizando com os clientes.

Resumindo, o intuito do trabalho visa mostrar que a qualidade quando possui metodologias com processos bem definidos minimiza os gastos futuros e poupa os problemas

e insatisfações futuras com o cliente. E, a associação do V-model com uma metodologia ágil, por exemplo, o SCRUM, facilita a gestão dos projetos e consequentemente a entrega do serviço de qualidade.

A pesquisa para chegar nessas conclusões foi de maneira exploratória, ou seja, realizadas através livros, internet, entrevistas realizadas com pessoas da área de QA e da área de gestão, artigos e monografias. Ao final do trabalho a seguintes pergunta será respondida: como pode ser minimizado/ reduzido a má qualidade no produto final (softwares em geral) e ao mesmo tempo não haja prejuízos mantendo a confiança e a credibilidade com o cliente?

Com isso, o trabalho será iniciado com uma breve história de como a qualidade foi inserida no nosso cotidiano e como ela é benéfica para as empresas de T.I nos dias de hoje. Após, será explicado como as ISOs, tipos, técnicas de testes estão inseridas no processo V-Model, no qual se tornam importante nos métodos para o sucesso de uma fábrica de teste. Para finalizar esse modelo em V será englobado ao SCRUM, no qual pode se tornar uma verdadeira aliada no processo de planejamento e agilidade na entrega do produto.

2 SOBRE A QUALIDADE

As empresas de hoje estão utilizando palavras como eficiência, agilidade e competitividade para satisfazer, ou mais superar, as exigências cada vez maiores dos seus clientes. O termo qualidade é o fator predominante quando se trata dessas exigências do mundo corporativo.

Para entender melhor como a qualidade está inserida nos dias de hoje uma análise deverá ser realizada para entender o motivo da criação dela. Com isso, a explicação da era do conhecimento será o cenário do começo de tudo, sendo esta uma era advinda da era industrial.

A era do conhecimento nada mais é que uma continuação da era anterior (industrial) sendo distinta referente ao enfoque. Isto é, o foco que a era industrial pregava era a mão de obra, já a era do conhecimento pregava as ideias, e com isso ocorreu uma revolução ocorrendo novas soluções, novas tecnologias.

O significado de tecnologia no grego é o estudo (logia) das técnicas / processos (tecno). Com o surgimento da tecnologia a era do conhecimento teve uma mudança muito importante, no qual os dados passam a ser tratados, ou seja, a uma transformação destes em informação.

Outro ponto importante é a transferência do conhecimento tácito para o explícito, ou seja, o conhecimento começa ser disseminado para outras pessoas através de livros, manuais, treinamentos entre outros que atualmente facilitam na distribuição de conhecimento entre as pessoas.

Porém, onde a qualidade entra nessa história? Com a era do conhecimento não só a rapidez e quantidade serão importantes, mas sim a qualidade do produto a ser entregue como mostrado a seguir.

Com as informações sendo disseminadas com agilidade, ainda mais com o advento da internet, as dúvidas dos clientes aumentava referente aos tipos, técnicas, modos que os serviços seriam realizados pela empresa contratada e como seu produto seria entregue. Com os anúncios das empresas oferecendo sempre o melhor produto com os melhores preços e melhor entrega ocorreu uma competitividade gigante entre as empresas do mesmo setor. E é

nesse ambiente que a qualidade nasceu, no qual o preço por um produto não seria mais apenas o valor e a agilidade, mas principalmente o quanto esse serviço e produto teriam de qualidade.

Para citar um exemplo simples para esclarecer essa ideia da qualidade, existe um comparativo de quando se compra algo de grande valor monetário, por exemplo, um carro. A primeira certeza que as pessoas possuem é que desejam comprar um carro de marca. E por que existe essa certeza? Porque elas (pessoas) sabem que aquela marca possui qualidade, ou seja, os carros possuem bons motores, acessórios, não existem reclamações com grandes frequências entre outras questões que podem ser levantadas. Isto é, mesmo que o carro da marca X seja mais barato e com maior agilidade de entrega, as maiorias das pessoas irão preferir o carro da marca que é conhecida e confiável.

Por isso, as empresas atuais estão em busca de serem “marcas” para atrair cada vez mais compradores tendo como item indispensável, a qualidade. Ela é um determinante que indica se a empresa será ou não uma “marca” para o mercado. Isso não seria diferente para o setor da tecnologia da informação (TI), pois os softwares / programas nada mais são que “carros projetados” para clientes que querem comprar algo específico para suas empresas.

Com isso, a qualidade pode ser definida como a capacidade de manter a estabilidade para agradar o cliente em que a qualidade oferecida, por exemplo, por uma consultoria de T.I. é a força que atrai esse cliente para a companhia. Então, a partir de toda essa explicação de como a qualidade é obtida há o crescimento e desenvolvimento de uma área no setor das fábricas de software, a área de teste cuja principal função é oferecer a melhor qualidade possível no seu produto – o software. E por que ela está em crescimento?

2.1 QUALIDADE NO SOFTWARE

Há pouco tempo atrás os softwares eram criados e testados pelos próprios desenvolvedores, ou seja, os testes eram simplicistas / básicos e realizados apenas para validação do código. Isto é, o negócio, o “como” o usuário iria utilizar e as várias vertentes de

erros que o usuário poderia cometer ficavam sem correções. E com isso, a alta taxa de reclamações, insatisfações; além de retornos desses softwares pelos clientes eram altas.

Até então, algumas fábricas de software continuavam a pensar que investir numa área de qualidade teria um custo muito elevado e com isso o preço de seus produtos iriam crescer e os clientes poderiam preferir outra empresa. Entretanto, isso não é a verdade como especificado na Regra de 10 de Myers (Figura 1) que estabelece que o custo de correção de defeitos tende a aumentar quanto mais tarde o defeito é detectado.

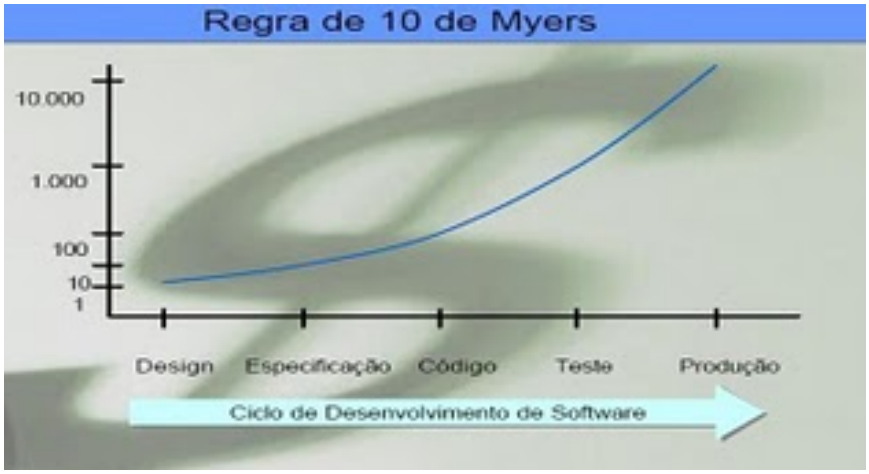

Figura 1: Regra de 10 de Myers (Fonte: http://qabrasil.blogspot.com)

Então nota-se que as empresas conseguem minimizar os custos quando os defeitos são encontrados no processo do desenvolvimento do que os encontrados em produção. No artigo “The cost of software quality” que o especialista Rex Black redigiu ele nos mostra através de uma planilha (Tabela 1) estimando os custos que as empresas sofrerão quando não há testes formais, quando há testes manuais e testes automatizados. Isto é, por meio da tabela (Tabela 1) se percebe que o retorno do investimento será maior à medida que o investimento em testes cresce.

INVESTIMENTO EM TESTES : ROI ANÁLISE (valores em dólares)			
	Não formal	Teste Manual	Teste Automatizado
Custo de Testes			
Pessoal	$ -	$ 60.000,00	$ 60.000,00
Infraestrutura	$ -	$ 10.000,00	$ 10.000,00
Ferramentas	$ -	$ -	$ 12.500,00
Investimento Total	$ -	$ 70.000,00	$ 82.500,00
Desenvolvimento			
Defeitos encontrados	250	250	250
Custo (defeitos)	$ 2.500,00	$ 2.500,00	$ 2.500,00
Testes			
Defeitos encontrados	0	350	500
Custo (defeitos)	$ -	$ 35.000,00	$ 50.000,00
Cliente			
Defeitos encontrados	750	400	250
Custo (defeitos)	$750.000,00	$ 400.000,00	$ 250.000,00
Custo de qualidade			
Conformidade (investimento em melhorias)	$ -	$ 70.000,00	$ 2.500,00
Não-conformidade (custo total dos defeitos encontrados)	$752.500,00	$ 437.500,00	$ 302.500,00
Total do Custo	$ 752.500,00	$ 507.500,00	$ 305.000,00
ROI	0%	350%	445%

Tabela 1: Estimativa de custos em testes (valores em dólares)

Com isso, os testes causam um impacto menor no "bolso" das empresas; além de minimizar problemas e danos quando a satisfação com o cliente e a qualidade do produto está em "jogo". Isso já estava sendo previsto no livro escrito por Wiley "The art of software testing" de 1979 em que ele afirmava o seguinte:

- Os testes unitários removem de 30% a 50% dos defeitos do produto.
- Os testes de sistemas removem de 30% a 50% dos defeitos remanescentes.

Isto é, mesmos com os testes o sistema ainda poderá ir para produção com 49%. Por último, há as revisões de código que reduzem entre 20% a 30% desses defeitos. O que se aproveita dessa informação? É que um sistema desenvolvido no século passado pode rodar gerações e continuar possuindo defeitos. À medida que esse programa vai ficando antiga a estatística sobre a idade do sistema prova que o custo da ocorrência dos defeitos vai ficando mais cara. Concluísse então que quanto mais defeitos são encontrados atualmente, menor o preço de manutenção que será gasto no futuro. Mais uma vez a diminuição de gasto pelas empresas pode ser mostrada através desses dados.

Percebe-se que uma fábrica de software nos dias atuais ganham vantagens competitivas e até mesmo financeira possuindo uma área de testes, mas mesmo assim deve existir uma estratégia, ou seja, é necessária a análise dos principais riscos para o negócio que uma determinada falha pode causar.

A importância de ter essa análise é para não criar testes desnecessários, ou seja, testes exagerados com minucias de detalhes que não interessam para o usuário, no qual poderão ocorrer gastos desnecessários de recursos. Com isso, há a necessidade de se estabelecer um ponto de equilíbrio para interrupções dos testes para o custo deste não supere o custo de falha para o negócio. Afinal, nenhuma empresa quer gastar a mais do que o essencial, ou melhor, do que o necessário para se obter qualidade com um bom preço.

Finalmente, concluísse esse tópico com três palavras chaves para entender o restante do trabalho: conhecimento, qualidade e lucro, no qual os interligando pode-se concluir que a qualidade em software veio através do conhecimento para gerar maiores lucros para empresas da tecnologia da informação que estão em constante crescimento.

Então, agora que o motivo da qualidade de software e sua história foram entendidos. Há a necessidade de se esclarecer como isso é normatizado, no qual a existência dessas normas ou padrões garanta a comprovação da qualidade dos softwares ditas no tópico acima.

2.2 NORMAS PARA QUALIDADE DE SOFTWARE

Para garantir que a qualidade do software esteja sendo empregada da maneira correta não basta só que ela exista, mas sim que ela seja comprovada. Com isso, existem as certificações que nada mais são que documentos oficiais indicando a conformidade com uma norma ou padrão que foi estimulado mundialmente sobre um item, no qual no nosso caso é a qualidade.

A existência de padrões ou normas, como a ISO (International Organization for Standardization); IEEE (Instituto de Engenharia Elétrica e Eletrônica); ABNT (Associação Brasileira de Normas Técnicas) garantam que essas certificações sejam emitidas à medida que as empresas padronizem seus processos de acordo com o que estiver estipulado em uma dessas normas.

O importante salientar que irão existir dois tipos de qualidades a serem padronizadas uma referente ao produto (software) e o outro referente ao processo. Abaixo na Tabela 2 constam as normas nacionais e internacionais que permitem a correta avaliação dos dois tipos da qualidade entre outros.

Norma	Comentário
ISO 9126	Características da qualidade de produtos de software.
NBR 13596	Versão brasileira da ISO 9126
ISO 14598	Guias para a avaliação de produtos de software, baseados na utilização prática da norma ISO 9126
ISO 12119	Características de qualidade de pacotes de software (software de prateleira, vendido com um produto embalado)
IEEE P1061	Standard for Software Quality Metrics Methodology (produto de software)
ISO 12207	Software Life Cycle Process. Norma para a qualidade do processo de desenvolvimento de

	software.
NBR ISO 9001	Sistemas de qualidade - Modelo para garantia de qualidade em Projeto, Desenvolvimento, Instalação e Assistência Técnica (processo)
NBR ISO 9000-3	Gestão de qualidade e garantia de qualidade. Aplicação da norma ISO 9000 para o processo de desenvolvimento de software.
NBR ISO 10011	Auditoria de Sistemas de Qualidade (processo)
CMM	Capability Maturity Model. Modelo da SEI (Instituto de Engenharia de Software do Departamento de Defesa dos EEUU) para avaliação da qualidade do processo de desenvolvimento de software. Não é uma norma ISO, mas é muito bem aceita no mercado.
SPICE ISO 15504	Projeto da ISO/IEC para avaliação de processo de desenvolvimento de software. Ainda não é uma norma oficial ISO, mas o processo está em andamento.

Tabela 2: Normas para qualidade do software. (Fonte: unemat.br)

Como o foco do trabalho não visa à explicação das normas, o foco será apenas em duas normas para maior entendimento da qualidade do produto e do processo, ISO 9126 e ISO 12207 respectivamente.

Para explicar as ISOs primeiramente devesse entender o significado da sigla ISO que em inglês é International Organization for Standardization, ou seja, organização internacional de normatização ou padronização. A ISO é uma organização não governamental fundada em Genebra (1947), no qual hoje está presente em 162 países tendo como principal finalidade promover a normatização de produtos e serviços, para que a qualidade dos mesmos seja permanentemente melhorada.

Com isso, serão citadas as duas ISOs que se tratam da normatização da qualidade do software conforme será visto nos tópicos 1.2.1 e 1.2.2.

2.2.1 ISO 9126

A ISO 9126 é uma norma para qualidade do produto de software sendo da família da norma 9000 que designa um grupo de normas técnicas estabelecendo um modelo de gestão de qualidade para organizações em geral.

Na ISO 9126 é estabelecido um modelo de qualidade com os seguintes componentes:

- **Processo de desenvolvimento:** cuja qualidade afeta a qualidade do produto de software gerado e é influenciado pela natureza do produto desenvolvido;
- **Produto:** compreendendo os atributos de qualidade do produto (sistema) de software. Estes atributos de qualidade conforme figura 2 podem ser divididos entre atributos internos e externos. Estes se diferenciam pela forma de como são aferidos (interna ou externamente ao produto de software) e em conjunto compõem a qualidade do produto de software em si;

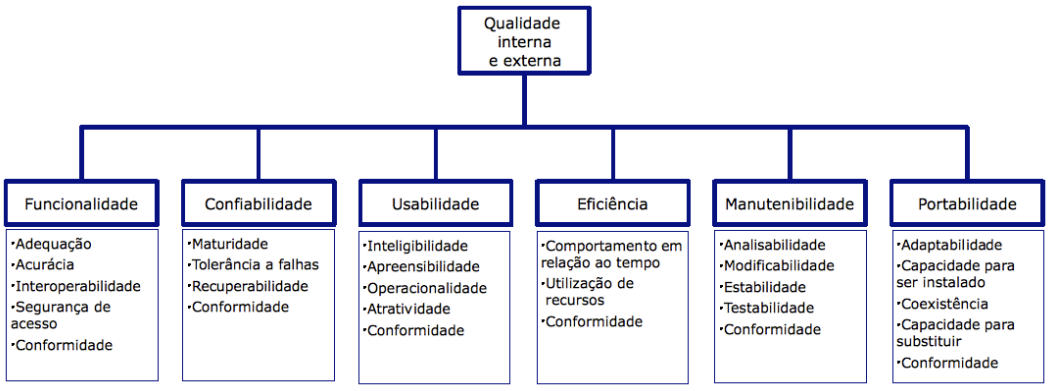

Figura 2: Atributo de qualidade (Fonte: http://pt.wikipedia.org/wiki/ISO/IEC_9126)

- **Qualidade em uso:** que consiste na aferição da qualidade do software em cada contexto específico de usuário. Esta é, também, a qualidade percebida pelo usuário.

Cada uma dos atributos de qualidade do software possui sua característica e sub-característica que deve ter nos sistemas criados pela empresa. Note que a sub-categoria “conformidade” aparece em todos os atributos, no qual verifica o quanto o sistema / software está obedecendo aos requisitos da legislação e da padronização proposta pela ISO.

2.2.2 ISO 12207

Diferente da norma 9126 a ISO 12207 define o processo do ciclo de vida dos softwares. Ela é a primeira norma internacional que descreve em detalhes os processos, atividades e tarefas que envolvem o fornecimento, desenvolvimento, operação e manutenção de produtos de software.

Os processos que essa norma detalha estão resumidos na tabela a seguir, no qual a metodologia de cascata V-model utiliza em todo o seu planejamento:

Processos Fundamentais	Início e execução do desenvolvimento, operação ou manutenção do software durante o seu ciclo de vida.
Aquisição	Atividades de quem um software. Inclui: definição da necessidade de adquirir um software (produto ou serviço), pedido de proposta, seleção de fornecedor, gerência da aquisição e aceitação do software.
Fornecimento	Atividades do fornecedor de software. Inclui preparar uma proposta, assinatura de contrato, determinação recursos necessários, planos de projeto e entrega do software.
Desenvolvimento	Atividades do desenvolvedor de software. Inclui: análise de requisitos, projeto, codificação, integração, testes, instalação e aceitação do software.
Operação	Atividades do operador do software. Inclui: operação do software e suporte operacional aos usuários.
Manutenção	Atividades de quem faz a manutenção do software.
Processos de Apoio	Auxiliam um outro processo.
Documentação	Registro de informações produzidas por um processo ou atividade. Inclui planejamento, projeto, desenvolvimento, produção, edição, distribuição e manutenção dos documentos necessários a gerentes, engenheiros e usuários do software.
Gerência de Configuração	Identificação e controle dos itens do software. Inclui: controle de armazenamento, liberações, manipulação, distribuição e modificação de cada um dos itens que compõem o software.
Garantia da Qualidade	Garante que os processos e produtos de software estejam em conformidade com os requisitos e os planos estabelecidos.
Verificação	Determina se os produtos de software de uma atividade atendem completamente aos requisitos ou condições impostas a eles.
Validação	Determina se os requisitos e o produto final (sistema ou software) atendem ao uso específico proposto.
Revisão Conjunta	Define as atividades para avaliar a situação e produtos de uma atividade de um projeto, se apropriado.
Auditoria	Determina adequação aos requisitos, planos e contrato, quando apropriado.
Resolução de Problemas	Análisar e resolução dos problemas de qualquer natureza ou fonte, descobertos durante a execução do desenvolvimento, operação, manutenção ou outros processos. .
Processos Organizacionais	Implementam uma estrutura constituída de processos de ciclo de vida e pessoal associados, melhorando continuamente a estrutura e os processos.
Gerência	Gerenciamento de processos.
Infra-estrutura	Fornecimento de recursos para outros processos. Inclui: hardware, software, ferramentas, técnicas, padrões de desenvolvimento, operação ou manutenção.
Melhoria	Atividades para estabeler, avaliar, medir, controlar e melhorar um processo de ciclo de vida de software.
Treinamento	Atividades para prover e manter pessoal treinado.

Tabela 3: Processos da Norma ISO 12207 (Fonte: unemat.br)

Após a explicação breve das ISOs envolvidas no modelo em V, o próximo capítulo será referente a metodologia em cascata V-Model que irá esclarecer todos os procedimentos envolvidos nessa área de testes para criar uma análise para criação de testes concisos e não desnecessários.

3 METODOLOGIA DE TESTES

O capítulo irá abordar sobre a metodologia em cascata utilizando o V-model ou modelo em V do ciclo de vida dos testes, no qual será explicado o que é a metodologia, os tipos, técnicas de testes; além dos procedimentos que devem ser utilizados para que o processo seja finalizado com sucesso e ganhe a confiança do cliente.

3.1 MODELO EM V OU V-MODEL

O modelo em V é um modelo conceitual utilizados em gestão de projetos que em 1980, com o surgimento da Engenharia de Software, se tornou um conceito padrão em todos os domínios da indústria de software.

Como dizia Iris Richter (analista sênior de testes), “Os benefícios advindos da adoção do modelo em ‘V’ (V-model) no âmbito de testes é detectar os defeitos precocemente, maior envolvimento do time de testes no início do projeto e aumento da qualidade do software”.

A principal ideia do V-model é testar sempre a mesma coisa, mas com focos diferentes. Para o entendimento do capítulos duas palavras devem ser conhecidas referente ao seu significado: validação e verificação. A diferença entre essas duas palavras pode ser respondida através das seguintes perguntas: Nós construímos corretamente o sistema? Nós construímos o sistema correto? A primeira pergunta diz respeito ao que foi construído, e a segunda ao entendimento do que era para ser construído, ou seja, a resposta da primeira pergunta seria a verificação enquanto a segunda a validação.

Resumindo, a verificação é fazer do jeito certo, ou seja, verificar se todo o processo foi realizado corretamente enquanto a validação é garantir que estão sendo feitas as coisas certas e notificar se tudo está correto para prosseguir adiante. Essas palavras serão utilizadas em

cada procedimento do início ao fim do projeto garantido maior qualidade na entrega de cada processo.

Além disso, o modelo em V possui dois “lados”, no qual no primeiro lado (direito) há o ciclo de vida de desenvolvimento e no outro lado (esquerdo) o ciclo de testes. Com isso, se verifica que os procedimentos de “fazer” e “conferir” (verificar e validar) convergem em todo o processo do modelo em V, conforme visualizado na figura 3.

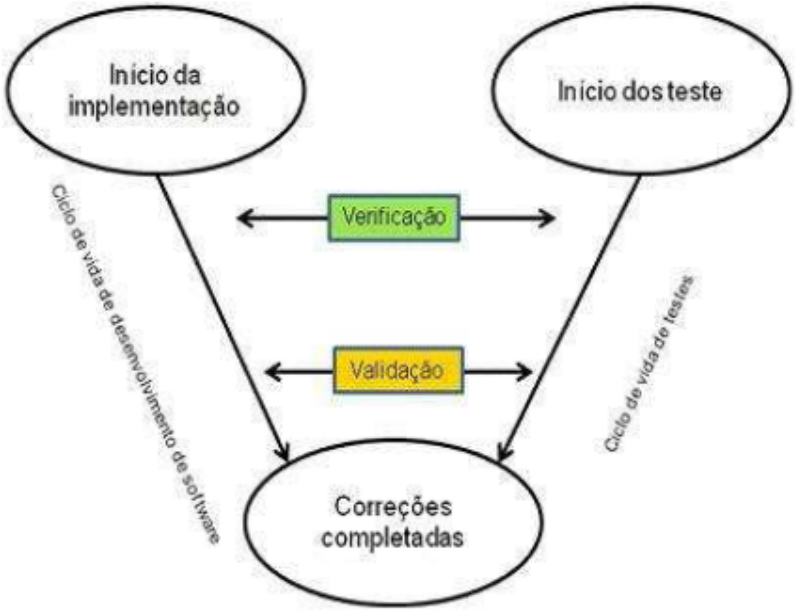

Figura 3: V-model (Fonte: Base de conhecimento em testes de software, 2007, pg. 41)

Esse modelo pressupõe que sejam realizados testes ao longo de todo o processo de desenvolvimento. Em determinados pontos, os produtos intermediários do ciclo de desenvolvimento são revisados para após isso ocorrer uma perfeita implementação procurando detectar o erro mais cedo possível.

O ciclo de vida de teste e de desenvolvimento são totalmente interdependentes, entretanto, o ciclo de testes depende da conclusão dos produtos e das atividades do ciclo de desenvolvimento. Essas atividades podem ser realizadas por grupos internos (profissionais da própria empresa para exercer as funções de teste) ou externos (pertencentes à outra empresa especialista em testes).

O importante é que o grupo de testes siga a metodologia e a cumpra para a produtividade e qualidade aumentar sempre distinguindo claramente quais são as funções e tarefas em cada etapa do ciclo. Além disso, as equipes de testes e desenvolvimento devem ser distintas, pois conforme algumas experiências realizadas é muito difícil um desenvolvedor realizar as duas funções (desenvolver e testar); além destes encobrirem seus próprios enganos de modo que a eficácia dos testes seja praticamente nula.

Com essa distinção é importante ressaltar que a metodologia de testes deve ser compatível com a metodologia de desenvolvimento. Essa compatibilidade se dá para alinhar as atividades de cada equipe viabilizando, o quanto antes, o começo dos testes e, com isso, a conclusão do projeto como um todo.

Segundo Anderson Bastos, no livro Base do conhecimento em teste de software, ele exemplifica os onze passos do software conforme a figura 4 e suas definições, no qual visualizasse uma visão macro do que ocorre no modelo em V.

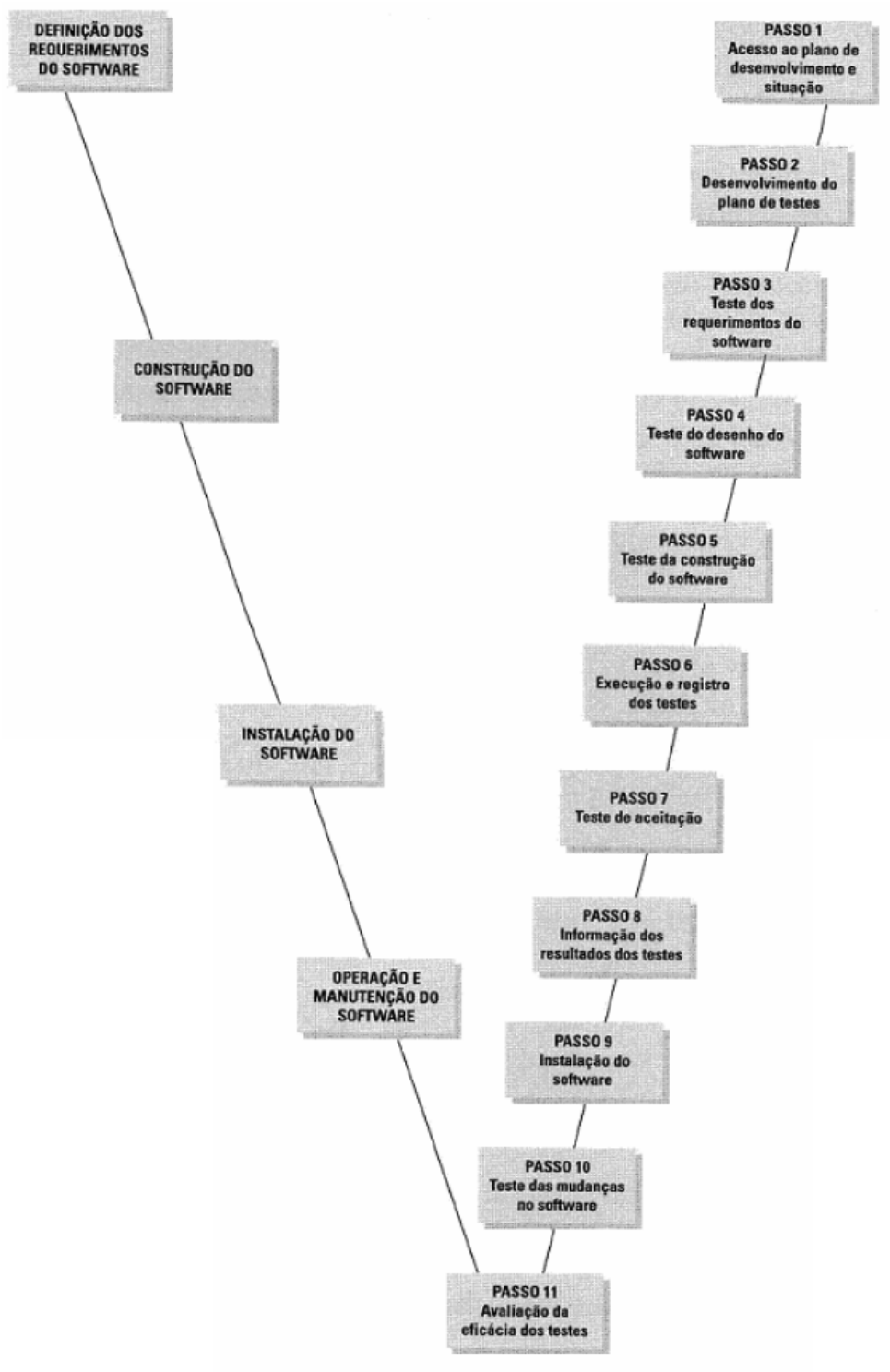

Figura 4: Onze passos do processo de software. (Fonte: Base de conhecimento em testes de software, 2007, pg.42)

Como dito anteriormente, visualizasse na figura 4 a parte direita referente ao desenvolvimento, no qual existem as partes de requerimento da aplicação, a construção (o desenvolvimento), a instalação e as operações e manutenção do software em relação aos erros encontrados. Já os processos referentes aos testes (lado esquerdo), existem os seguintes tópicos explicados a seguir:

1. Acesso ao Plano de Desenvolvimento: Pré-requisito para a construção do plano de teste. Esse é o passo que os testers (testadores) irão verificar se o plano de desenvolvimento está correto, no qual estimasse a quantidade de recursos utilizada no projeto de teste.

2. Desenvolvimento do Plano de Teste: planejamento do Plano de teste, no qual serão verificados os riscos que a aplicação pode ter, no qual o objetivo é minimizar os possíveis erros. Nessa fase serão criados os cenários de teste com os seus passos-a-passos da execução e os resultados obtidos esperados sendo por meio dele a o veredicto de sucesso ou falha.

3. Inspeção ou teste de requisitos: esse teste é a avaliação referente a todos os requisitos realizados por meio da técnica de verificação, no qual problemas neste geram insucesso para a boa parte do desenvolvimento de software.

4. Inspeção ou teste do desenho de software: esse teste é parecido com o passo 3, entretanto, muda o foco para o desenho (interno e externo) do software, no qual verificasse este está condizente com os requerimentos.

5. Inspeção ou teste da construção do software: nesse passo será validado a extensão que o teste terá a partir da visualização de como a aplicação foi construída a partir do desenho.

6. Execução dos testes: envolve testar o código em estado dinâmico. A abordagem, as ferramentas e os métodos especificados no Plano de Teste serão empregados nas validações dos requisitos, desenhos e códigos. Os tipos de testes que devem ser usados para cada fase será explicado no próximo tópico.

7. Teste de aceitação: avaliação do usuário referente a usabilidade e aplicabilidade do software. Nesse passo verificasse a expectativa do cliente, no qual poderá gerar mudanças

no software de acordo com a sua necessidade; além de erros referentes a não compreensão dos requisitos. Essa etapa mostrará se o software poderá ou não ser implantado.

8. Informação dos resultados dos testes: esse passo refere-se à informação dos resultados obtidos dos testes para que as equipes envolvidas tenham como preferência o aviso formalizado, ou seja, escrito e não o informal falado.

9. Teste da instalação do software: verifica a interoperabilidade com o sistema operacional entre outras aplicações que podem causar impactos no novo software da empresa (aplicação que está sendo testada).

10. Teste das mudanças no software: as atividades desse passo é verificar as mudanças durante o processo de implementação e aquelas que vão ocorrer após a implantação.

11. Avaliação da eficácia dos testes: o último passo é verificar o quão eficaz foi o teste e pode envolver diversas áreas além de QA como desenvolvimento e o próprio usuário. Essa fase é importante, pois pode haver uma pesquisa de como o processo pode ser melhorado dependendo da avaliação final.

Para a realização dos testes realizasse os seguintes procedimentos para cada um desses onze passos, no qual será utilizado o modelo 3P x 3E conforme o próximo tópico.

3.2 MODELO 3P X 3E

O ciclo de vida do processo de teste é composto por diversas fases e etapas sendo quatro delas em cascata, e duas em paralelo. As duas em paralelo são planejamento e preparação que participam ao longo do processo fornecendo suporte as outras fases. Os procedimentos iniciais é a fase mais curta do processo, no qual 80% a 85% deste (processo) vai se concentrar no restante, ou seja, especificação, execução e entrega. Esse modelo é chamado de 3P e 3E conforme mostrado abaixo na figura 5.

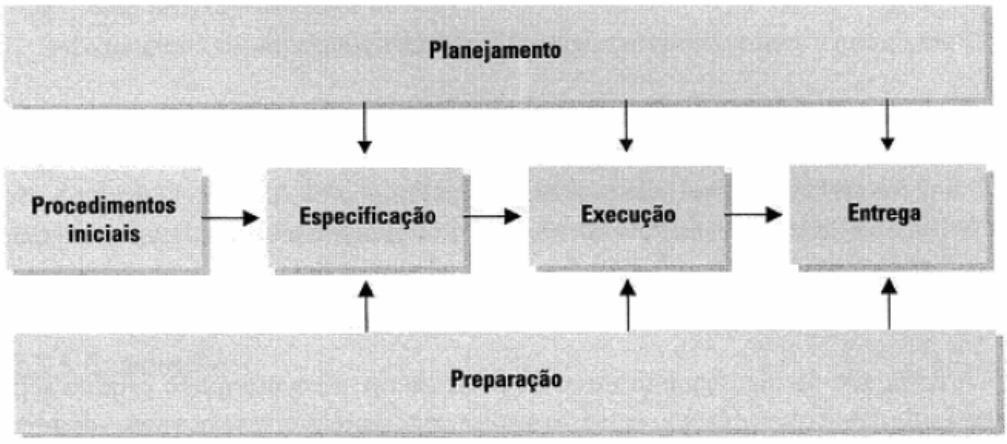

Figura 5: Modelo 3P x 3E (Fonte: Rios, E.& Moreira, T. Teste de software. Rio Janeiro, Alta Books, 2003.)

As atividades, produtos e documentos atrelados a cada fase pode ser resumido conforme a seguir:

- **Procedimentos iniciais:** é a abordagem inicial, no qual visualizasse a estratégia de teste, o *work plan*. Isto é, será elaborado o plano com todas as atividades principais que serão executadas – se possível, com necessidades de recursos (pessoal e ambiente). Nesse nível será possível ser elaborado o GOT (Guia Operacional de Teste), um documento básico do acordo de nível de serviço entre as áreas atuantes.

- **Planejamento:** consiste em elaborar a estratégia e o plano de teste visando minimizar os principais riscos do negócio e fornecer caminho para as próximas etapas. Nessa etapa é onde serão analisados os requisitos, ou seja, se perguntar o que testar? Como testar? Como integrar? Além de realizar o planejamento do projeto (horas, pessoas e etc).

- **Preparação:** essa fase tem como objetivo a preparação do ambiente, pessoal, equipamentos, ferramentas de automação entre outros para que os testes sejam realizados com sucesso.

- **Especificação:** o objetivo é elaborar / revisar os casos de testes, no qual à medida que novas versões são liberadas pela equipe desenvolvimento, a equipe de QA (quality assurance) – qualidade – devem elaborar os casos e roteiros dinamicamente, ou seja, juntamente com essa atividade.

- **Execução:** como o próprio nome diz é a etapa de executar os casos de testes, entretanto, é necessária criar diretrizes para essa execução, como: seguir os casos de testes e os roteiros, sempre colher evidências dos testes que estão sendo executados. À medida que surge uma nova versão, sempre realizar testes de regressão que possibilitam “retestar” e verificar que tudo da versão antiga ainda funciona nessa versão atual. Enfim, essa é a fase será necessário se atentar aos possíveis erros que ocorrerem e reportarem imediatamente para o time de desenvolvimento realizar a correção dos mesmos.

- **Entrega:** a etapa que finaliza o processo e que arquiva toda a documentação referente ao teste. Essa é a etapa onde serão concluídos / finalizados os testes.

Enfim, todas essas etapas deverão existir em cada fase de teste do V-Model sendo utilizada em todo processo. Entretanto, para se entender o modelo em V por completo será preciso ainda explicar teoricamente os tipos de testes, suas técnicas e ferramentas para se juntar todos esses tópicos no estudo de caso da fábrica de software (2.6) e assim sanar possíveis dúvidas sobre essa metodologia. Com isso, será mostrado primeiramente os tipos de testes conforme o tópico 2.3.

3.3 TIPOS DE TESTES

A realização dos testes deve refletir as ações tomadas para descobrir os erros de código que as funcionalidades do sistema possam conter entre outros erros de especificação. Como já explicado anteriormente o V-model consiste em realizar os mesmos testes, mas com diferentes focos. Com isso, a utilização do V-model é primordial para a área de testes, pois ele testará todos os níveis / estágios do sistema como um todo; além de verificar o “quando”, ou seja, a que fase do desenvolvimento se aplica determinado teste, no qual existem os testes: unitário, de integração, de desempenho (performance), do produto e o de aceitação que serão explicados a seguir:

- **Teste unitário ou componente:** esse é o primeiro teste que normalmente são realizados pelos desenvolvedores, no qual existem algumas ferramentas que auxiliam nesse tipo de teste criando eficácia e rapidez em todo o processo. Como o próprio nome diz, unitário, esse teste é realizado na unidade do sistema, ou seja, na menor parte da aplicação que seria uma função, procedimento do próprio código.

- **Teste de integração:** esse teste deve ser realizado pelo analista de teste bem como todos os outros adiante, no qual este se refere à integração entre componentes, ou seja, as unidades testadas no teste unitário (módulos) são integradas para criar a aplicação por completo. Com isso, esse teste é necessário para validação dos módulos, ou seja, ao "ligar" todas as unidades da aplicação o sistema funcione integralmente. É importante observar que ele não testa a funcionalidade da aplicação, mas sim a parte técnica (código) ainda.

- **Teste de performance / desempenho:** esse teste se caracteriza para avaliar o desempenho do sistema, como perfis de andamento, fluxo de execução, tempos de respostas, confiabilidade e limites operacionais.

- **Teste do produto:** esse é o teste que irá verificar todas as funções que a aplicação deve exercer. O ambiente de teste deve ser o mais próximo possível do de produção para que os resultados sejam confiáveis. Aqui todos os cenários testes criados referentes aos requisitos do cliente serão executados e avaliados. Na eminência de um defeito a equipe de desenvolvimento será notificada para a correção dos mesmos. Essa é a fase de teste mais longa devido à verificação de cada funcionalidade do sistema.

- **Teste de aceitação:** o teste que finaliza a fase do V-model, no qual é realizado pelos usuários e testadores para garantir que tudo o que foi definido nos requerimentos tenha sido incluído na aplicação. Nessa fase podem ocorrer mudanças que serão retestadas em próximas versões e/ou novos projetos.

Os tipos de testes sempre devem procurar melhorar continuamente; como realizar a evolução dos processos e das equipes relacionadas para assim garantir maior qualidade do trabalho. Para isso, existem técnicas de testes que nos auxiliam na criação dos cenários bem como as execuções destes.

3.4 TÉCNICAS DE TESTES

Para realizar os testes descritos acima há algumas técnicas de testes, no qual se dividem em dois tipos: testes estruturais e testes funcionais. Os testes estruturais são os que garantem atendimento aos requisitos enquanto os estruturais garantem o contexto técnico onde serão instalados, como o funcionamento de todos os aspectos da estrutura.

3.4.1 TÈCNICAS DE TESTE ESTRUTURAL

Como explicado anteriormente essa técnica irá determinar se a tecnologia foi utilizada de modo adequado e se os componentes / módulos montados funcionam de maneira coesa. Os tipos de testes que irão utilizar essas técnicas serão os testes unitários e os de integração, no qual será explicada cada uma a seguir.

- **Teste de estresse:** o objetivo é avaliar o comportamento da aplicação sob condições críticas, como: verificar o volume de transações que a aplicação poderá suportar; verificar se a estrutura do sistema irá comportar grandes volumes de dados, se o tempo de resposta da aplicação está condizente com o solicitado pelo cliente entre outros. A ideia desse teste é “estressar” o sistema para verificar até que ponto ele poderá chegar e se isso estará coeso com os requisitos que o cliente solicitou.

- **Teste de execução:** este teste é empregado para validar os critérios de desempenho da aplicação, ou seja, o teste irá determinar o desempenho da estrutura do sistema como o tempo de processamento das transações; além de verificar o nível de utilização do hardware e do software. Essa técnica será muito utilizada no teste de performance para validar que o desempenho do sistema esteja funcionando como o esperado.

- **Teste de recuperação (contingência):** esse teste é realizado quando existe um backup para a aplicação testada, no qual garanta a continuidade das operações após um desastre; além de verificar a eficácia das partes componentes do processo.

- **Teste de operação:** o objeto é validar que o sistema será executável durante a operação normal, ou seja, após os testes da aplicação deverá haver esse teste para verificar a integração com o sistema operacional antes do software entrar em produção.

- **Teste de conformidade:** esses testes são realizados para garantir a conformidade com as metodologias e encorajar e auxiliar os profissionais de TI a adotá-las.

- **Teste de segurança:** estes são necessários para garantir a confidencialidade das informações e a proteção dos dados. A segurança da aplicação irá depender dos riscos associados, no qual serão identificadas as possíveis falhas para manter a confidencialidade do usuário em relação a esse software.

3.4.2 TÉCNICAS DE TESTE FUNCIONAL

Os testes funcionais são utilizados para validar os requisitos e especificações da aplicação, ou seja, serão verificadas as técnicas que serão usadas para garantir que todos os requisitos realizados pelo cliente estejam contidos dentro da aplicação. Os tipos testes que utilizam mais essa técnica são os de produto, performance (desempenho) e o de aceitação como explicado anteriormente. Sendo assim, segue abaixo as técnicas para esses tipos de teste:

- **Teste de requisitos:** como o próprio nome diz os testes serão em cima dos requisitos cuja finalidade é verificar se as funcionalidades da aplicação estão condizentes com o sistema e se este será capaz de sustentar correções após sua utilização por um período de tempo contínuo.

- **Teste de tratamento de erros:** esses testes irão verificar a capacidade do sistema de tratar as condições de erros. Isto é, verificar se todas as condições de erros serão

reconhecidas pelo sistema; determinar se foi atribuída responsabilidade para processar esses erros; manter um controle sobre esses erros durante o processo de correção.

- **Teste de interconexão:** esse tipo de teste é utilizado quando há uma conexão com outros softwares podendo essa conexão ser através de recebimento, envio de dados entre uma ou ambas as aplicações. A ideia é garantir que a interconexão funcione corretamente, no qual os dados serão verificados referente sua transferência coerente, como o momento certo de execução e a existência da coordenação das funções entre os softwares.

3.4.3 OUTRAS TÉCNICAS

Além das técnicas abordadas acima existem outras nomenclaturas de técnicas muito utilizadas na área de testes que são muito usadas pelos testadores de software, no qual é de extrema importante conhecê-las que são:

- **Teste caixa preta:** é uma técnica que verifica a saída dos dados usando entrada de vários tipos. O importante desse teste é que se conhece a entrada e a saída, mas se desconhece o código da aplicação. Isto é, essa técnica é testar as funcionalidades e não o técnico (código) em si.

- **Teste caixa branca:** é uma técnica inversa a da caixa preta, no qual usa a perspectiva interno do sistema para modelar o caso de teste. Nesse teste se conhece o código e o testador necessita de *skills* (habilidade) em programação para realização dos testes.

- **Teste positivo:** esse teste visa validar testes positivos, ou seja, tudo o que deveria funcionar na aplicação tem que estar funcionando e o que não deveria funcionar não está funcionando. Como alguns testadores explicam esse é o teste para passar.

- **Teste negativo:** contrário do teste positivo esse é o teste de exceção, teste para falhar. Isto é, esse teste tem que validar as exceções da aplicação gerando um erro, como por exemplo, a mensagem de uma senha inválida.

- **Teste estático:** nesse tipo de teste o código é examinado, ou seja, os testes são realizados “dentro” do código.
- **Teste dinâmico:** diferente do teste estático ele testa a funcionalidade dos sistemas com valores de entrada e saída.

Para todos os tipos e técnicas de testes há necessidade de maneiras para agilizar todo o processo. Com isso, há no mercado as ferramentas de teste que será mostrada no tópico 3.5.

3.5 FERRAMENTAS DE TESTE

Todos esses tipos e técnicas de testes poderão ser realizados automaticamente, por meio de ferramentas. Entretanto, é importante esclarecer que todo testador deve conhecer primeiro as técnicas de teste para depois saber quais ferramentas deverão ser usadas com cada uma.

Enquanto a técnica é um processo que assegura o funcionamento de alguns aspectos do sistema, a ferramenta é um veículo para executar os processos do teste sendo importante salientar que a ferramenta é um recurso para o tester (testador), e que sozinha ela se torna insuficiente para realizar todo o teste. Isto é, sem uma técnica a ferramenta nem sempre será eficaz.

Tanto a atividade de teste quanto a de desenvolvimento geram um grande número de informações que necessitam ser repetidas várias vezes. Com isso, as ferramentas de testes podem aliviar esse fardo da produção e execução das informações.

A maioria dos profissionais de TI de testes utilizam as ferramentas como apoio utilizando umas das técnicas de testes, no qual é importante lembrar que algumas técnicas só são utilizadas manualmente, ou seja, não poderão ser automatizadas pelas ferramentas.

As técnicas que serão automatizadas devem encontrar uma ferramenta adequada nos diversos tipos existentes que variam em relação ao conhecimento e custo sendo algumas

extremamente técnicas requerendo *skills* de programação e outras mais genéricas utilizadas por quase todos os profissionais da área de teste.

Com isso, após os casos de testes serem realizados pode-se realizar a automatização destse sendo sempre atrelado a uma técnica de teste. Para isso, devesse escolher uma ferramenta adequada e realizar a criação dos scripts que nada mais é que o passo-a-passo que a ferramenta terá que realizar para concluir os testes sendo realizada de diversas formas podendo o tester gravar ou programar. Além disso, há a necessidade de se criar a massa de dados que são os dados de entrada que deve ser informado na aplicação para poder funcionar, no qual pode ser o nome de um usuário, senha entre outros. Assim, após os dados de entrada, será verificado os dados de saída, os resultados obtidos como mensagem de erro ou sucesso entre outros.

O processo de informar um dado de entrada, no qual resultará em um dado de saída é o que a ferramenta de teste validará inserindo o veredito / status de sucesso ou falha. Há ferramentas que são integradas com outras ferramentas, como, por exemplo, existem ferramentas que são apenas para armazenamento e gerenciamento de casos de testes, e outras que realizam as execuções dos scripts. Quando esta ferramenta executar os scripts e identificar o status (sucesso ou falha) ela já informará o status a outra ferramenta facilitando os gestores dos projetos de testes na criação dos status gerenciais.

Como em qualquer área de T.I sempre haverá ferramentas sendo criadas e evoluídas para atender as demandas de serviços e isso não seria diferente na área de testes, no qual estas diariamente facilitam as vidas dos testadores executando scripts e gerando resultados para uma melhor qualidade e rapidez na entrega do software.

3.6 ESTUDO DE CASO - MODELO EM V EM UMA FÁBRICA DE SOFTWARE

Nesse tópico serão englobados todos os procedimentos, processos, técnicas citadas acima a fim de esclarecer a metodologia V-model mostrando na prática o cotidiano em uma fábrica de software. Nesse tópico serão estudados todos os passos da Figura 6, no qual demonstra o modelo em “V” da fábrica de software.

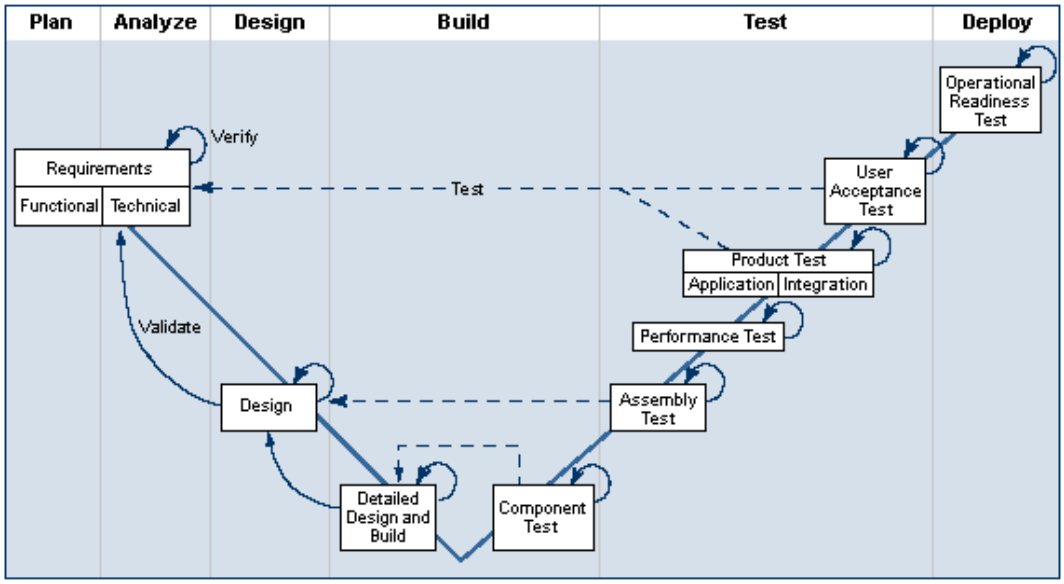

Figura 6: V-model (modelo em V)

Na parte superior da Figura 6, podem ser visualizadas as cinco fases, no qual se inicia pelo *Plan* (Planejamento) que é a fase da criação da estratégia que junto com a posterior *Analyse* (Análise) realiza o processo de estudar os requisitos e transformá-los em cenários de testes. Já a fase *Design* (Desenho / Projeto) é a parte onde a aplicação é projetada, ou seja, as respostas para as perguntas: Como o sistema será feito? Quais serão os requisitos dessa aplicação? O que queremos que essa aplicação realize? Após, a fase *Build* (Construção) que é onde a aplicação será construída / programada a partir do que foi planejado nas fases anteriores.

A penúltima fase é o foco será empregado, pois é a fase *Test* (Teste), nos quais haverão os tipos de teste para cada "etapa", no qual já foi estudado sua teoria em tópicos anteriores. Por último haverá a fase de *Deploy* (Implantação), que nada mais é que a instalação da aplicação no ambiente da empresa concluindo o desenvolvimento e os testes finalizando o modelo em V.

Como mostrado acima, há uma semelhança entre essas fases com as do modelo 3P X 3E, mesmo tendo uma igualdade nas definições deve ser ressaltado que todos os procedimentos referentes ao modelo 3P x 3E devem ser realizados para cada um dos processos que são representados pelas "caixinhas" como mostra a Figura 6, ou seja, todo processo do modelo em V deve conter as tarefas que foram ditas sobre o modelo 3P X 3E. Já as fases que estão acima na Figura 6 são as fases em que os processos serão realizados.

Resumindo, cada processo (“caixinha”) deve contemplar o modelo 3P X 3E, no qual estes estão contidos em fases do V-model como um todo. Para esse contexto ser compreendido será explicado os processos do V-model começando pelo lado esquerdo, ou seja, ciclo de desenvolvimento como já visto no tópico 2.1, no qual será explicado detalhadamente o funcionamento dos procedimentos que devem ser realizados.

O primeiro processo é o *Requirement* (Requerimento) cuja sua função é verificar os requerimentos da aplicação do cliente estando por isso contido nas fases de planejamento e análise. A empresa desse estudo de caso dividiu os requerimentos em dois tipos: enquanto a parte *functional* (funcional) comtempla as funcionalidades básicas (login, telas) a parte *technical* (técnico) verifica as suas restrições dos sistemas (tempo de abertura, segurança) podendo ser realizada por diferentes profissionais dependendo do seu perfil mais técnico ou mais funcional.

Ainda nesse processo visualizando a Figura 6 é possível verificar as duas palavras de importância para modelo em V: *“verify”* (verificação) e *“validate”* (validação). Isto é, a idéia do V-model sempre realizar o “fazer” e “conferir” para todos os processos. Nessa fábrica de software essas duas palavras serão “transformadas” em documentações necessárias para todos os processos que são o *“Criteria Entry” e o “Criteria Exit”*.

Os critérios de entrada e de saída são conjunto de condições que devem ser satisfeitas antes de entrar e sair de um processo do V-model. O estado de critérios que é necessário, por exemplo, de estágios anteriores para suportar a próxima etapa seria referente aos critérios de entrada (verificação), já o que é exigido do critério de saída é a determinação da integralidade para um próximo estágio (validação). Esses critérios de entrada e saída são definidos para cada etapa para garantir resultados de qualidade de um processo para o próximo.

O modelo em V especifica que as atividades de uma fase só poderão ser iniciadas após as outra serem concluídas. Com isso, é importante salientar que os procedimentos de critérios de entrada e a saída serão utilizados em todo o processo V-model, ou seja, em cada “caixinha” visualizada na Figura 6. Sendo assim será passado para o próximo processo que seria o Design (Desenho), no qual só poderá ser iniciado com a aceitação dos dois documentos de *Criteria* da fase anterior *Requirement.*

O processo *Design* (Desenho) seria onde a aplicação seria planejada / estruturada, no qual se encontra na fase com o mesmo nome; além disso, verificasse que há uma convergência do *Assembly Test* (Teste de integração) que será explicado adiante.

A próxima fase é a da *Build* (Construção) e com ela há dois processos, o primeiro relacionado ao desenvolvimento da aplicação e detalhes finais do desenho, o *Detailed Design and Build* (Detalhes do Desenho e Construção). E o outro relacionado à parte de testes, *Unit Test* (Teste unitário). O interessante é visualizar que estes dois processos estão na mesma fase e eles se "convergem", pois como o teste unitário testa a mínima parte da aplicação, como explicado no tópico 2.3, este deverá estar condizente com todos os itens da fase anterior (*Detailed Design and Build*), no qual em sua maioria as execuções destes serão realizadas pelos próprios desenvolvedores.

Migrando para o lado direito do modelo em V e também para a fase de *Test* (Testes) será iniciado o fluxo de todos os tipos de testes que foi comentado no tópico 2.3. iniciando pelo *Assembly Test* (Teste de integração) que está intimamente relacionado com o processo *Design* (Desenho) da fase de desenvolvimento. Isto significa que esse teste deverá ser realizado a partir das especificações que a fase Design documentou, ou seja, todos os cenários de integrações dos componentes deverão constar no desenho da aplicação para os testes serem efetivos.

O próximo processo é o *Performance Test* (Teste de Desempenho), no qual testará o desempenho do sistema. O importante é que dependendo do tipo da aplicação ou projeto este teste poderá não ser realizado devido ao custo elevado e a não necessidade desse tipo específico de teste. Caso não ocorra a realização do teste de desempenho será adiantado para o próximo processo o *Product Test* (Teste do Produto).

O teste de produto é o principal teste quando se trata de funcionalidades do sistema, pois serão nestes que os erros que o usuário / cliente encontram em seu sistema. Esse teste está dividido em duas vertentes: Application (aplicação) ou Integration (integração), no qual sua diferença é que no teste da aplicação tratam-se os das funcionalidades do software como as telas da própria aplicação, já os da integração são realizados quando há uma aplicação externa e são feitos os testes entre esta e a aplicação a ser desenvolvida e, por exemplo, um banco de dados, outros sites e etc.

O importante é notar a diferença entre o teste de integração e o teste de produto integrado, no qual enquanto o teste de integração testa componentes internos, ou seja, módulos da mesma aplicação, o teste integrado do produto são testes da aplicação com componentes externos.

Após o final do teste do produto pode ser visualizado o último processo da fase de testes, o *User Acceptance Test* (Teste de aceitação), no qual o usuário será envolvido e a aplicação será validada junto com a equipe de teste e desenvolvimento. Nos dois últimos processos, teste do produto e aceitação, mostrasse a convergência com o processo de requisitos (*Requirement*). Isto significa, que esses dois processos da fase de teste deve conter todos os requisitos conforme descrito no processo da fase de desenvolvimento para comtemplar todas as solicitações do cliente fez em relação a seu software.

O último processo é o teste final para a entrega da aplicação para o cliente, o *Operational Readiness Test* (Teste de aptidão operacional) que está contido na fase de *Deploy* (Implantação). Este será o último teste realizado no cliente para certificar que nenhum problema de ambiente possa vir ocorrer. Normalmente os testadores já realizam os testes no ambiente do cliente para evitar que esse problema aconteça.

A empresa de fábrica de teste é uma multinacional que comprova que a utilização do V-model é de fundamental importância para os funcionamentos das atividades tanto de desenvolvimento quanto a de testes.

Concluísse assim que essa metodologia quando bem empregada ajuda ainda mais na qualidade dos produtos desenvolvidos devido à utilização de processos bem definidos, procedimentos concisos, tipos e técnicas de testes adequadas e documentações escritas evidenciando cada etapa dessa grande metodologia.

Bom, explicada a metodologia em cascata utilizando o V-model, mas para criar uma maior agilidade nesse processo será necessário englobá-la junto com uma metodologia ágil (SCRUM) que apesar de serem diferentes podem ser aliadas nos projetos de softwares conforme no próximo capítulo.

4 V-MODEL E SCRUM

Neste capítulo será abordada a metodologia SCRUM com a metodologia V-model no intuito de melhorar ainda mais os benefícios de possuir uma área de testes nas empresas; além de mostrar que a metodologia em cascata pode se unir a outras metodologias para proporcionar melhorias num processo como um todo.

Com isso, será mostrado nesse tópico como uma metodologia ágil pode melhorar a entrega e o planejamento das tarefas acelerando os processos do V-model de maneira rápida e prática. Entretanto, para isso, será preciso explicar como a metodologia ágil SCRUM funciona e quais são seus principais conceitos.

4.1 METODOLOGIA SCRUM

Com o a criação do Manifesto Ágil, ou seja, a declaração dos princípios que fundamentam o desenvolvimento ágil de softwares ocorreu o surgimento das metodologias ágeis que são utilizadas desde pequenas a grandes empresas na atualidade. Esse manifesto possui alguns princípios, no qual as metodologias se baseiam que são:

- Garantir a satisfação do consumidor entregando rapidamente e continuamente softwares funcionais.
- Softwares funcionais são entregues frequentemente (semanas, ao invés de meses).
- Softwares funcionais são a principal medida de progresso do projeto.
- Até mesmo mudanças tardias de escopo no projeto são bem-vindas.
- Cooperação constante entre pessoas que entendem do 'negócio' e desenvolvedores.

- Projetos surgem através de indivíduos motivados, e que deve existir uma relação de confiança.
- Design do software deve prezar pela excelência técnica.
- Simplicidade.
- Rápida adaptação às mudanças.
- Indivíduos e interações mais do que processos e ferramentas.
- Software funcional mais do que documentação extensa.
- Colaboração com clientes mais do que negociação de contratos.
- Responder a mudanças mais do que seguir um plano.

Além dos princípios há a necessidade de entender os quatro valores fundamentais para desenvolver um software ágil, que são: os indivíduos e suas interações acima de procedimentos e ferramentas; o funcionamento do software acima de documentação abrangente; a colaboração dos clientes acima da negociação de contratos; a capacidade de resposta a mudanças acima de um plano pré-estabelecido.

Assim se “cria” os fundamentos para uma metodologia ágil como o SCRUM. Após serem mostrados quais são os valores e princípios que esta está envolvida será necessário resumir em alguns fatores que será utilizado como comparativo entre metodologias: simplicidade, agilidade nas decisões, novidades sempre são bem vindas e rapidez na entrega. Mas antes de realizar a associação do SCRUM com o V-model há a necessidade de entender o que é o SCRUM e como ele é usado nas empresas em geral.

A ideia do SCRUM é ser uma metodologia ágil para gestão e planejamento de projetos de software, no qual estes são divididos em ciclos (tipicamente mensais) chamados *Sprints*. Entretanto, antes de iniciar as explicações dos conceitos será explicado quem é quem no SCRUM, com isso, segue a definição de cada pessoa nessa metodologia.

- Scrum Master: ele realiza as mesmas funções de um gerente de projeto ou a um líder técnico. Ele possui a responsabilidade de ser um a facilitador, ou seja, remove os

obstáculos que a equipe possa ter; além de assegurar que ela não se comprometa excessivamente com aquilo que é capaz de realizar durante um *sprint.*

- Product Owner (proprietário do produto): é a pessoa que defini os requisitos que serão entregues no *sprint* se comprometendo a entregá-lo na data esperada. O nome desses requisitos de cada *sprint* se chama *product backlog,* no qual o *Product Owner* se junta ao *SCRUM Master* para definir as listas de funcionalidade desejadas para um produto (software).

- Scrum Team (Equipe): é um grupo multifuncional normalmente de até sete pessoas que executam as tarefas dos *sprints* ditas pelo *Product Owner*.

É importante ressaltar que cada *sprint* é uma iteração que segue um ciclo PDCA (Plan Do Control Act), no qual ao final do processo é realizada a entrega de uma parte do software. Além disso, após a definição do *Product backlog* (requisitos do produto) é realizada uma reunião, no qual participa todos os integrantes para definição do *Sprint Backlog*, ou seja, os requisitos para aquela iteração.

Para manter as informações alinhadas sobre o projeto é realizada a *Daily Meeting* (Reunião diária), no qual são breves reuniões para disseminar conhecimento sobre o que foi feito e de possíveis problemas que possam estar impactando o projeto.

Ao final de cada *sprint*, a equipe apresenta as funcionalidades implementadas, no nosso caso, os testes que foram realizados em uma reunião chamada *Sprint Review Meeting*. (Reunião de revisão da iteração). Além disso, como todo processo pode ser melhorado há uma reunião onde todos podem colocar os pontos positivos e negativos e sugerir melhorias no *Sprint Retrospective* (Retrospectiva da iteração).

Abaixo na figura 7 segue o ciclo do SCRUM resumido referente aos conceitos explicados nesse tópico:

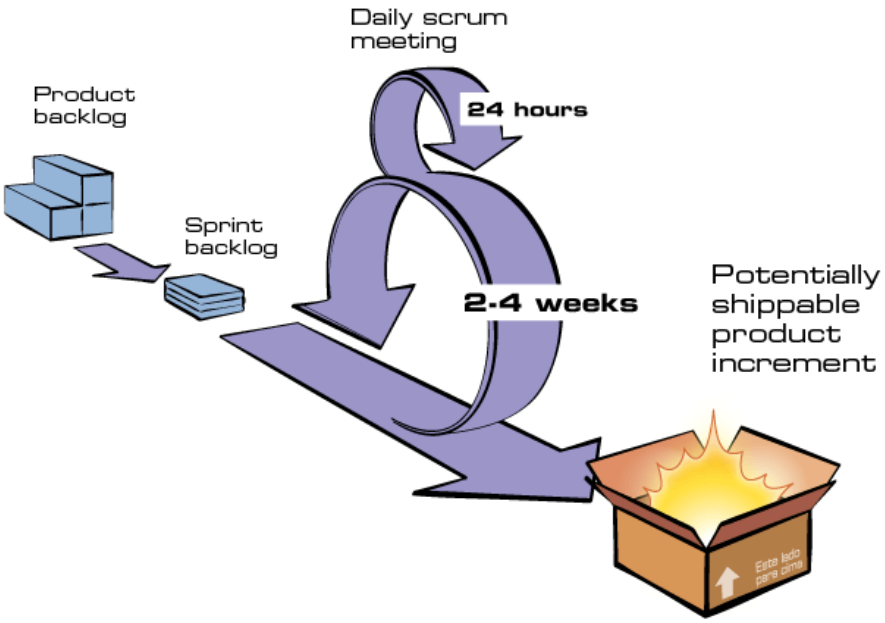

Figura 7: Ciclo Scrum. (Fonte: http://improveit.com.br/scrum)

Com isso, a explicação sobre SCRUM foi finalizada, mas deixando uma pergunta: será possível unir a metodologia tradicional V-model com a metologia ágil apresentada? A simplicidade ou burocracia das documentações? Os processos detalhados ou as mudanças repentinas? Equipes gigantes com funções predeterminadas ou equipes multifuncionais? Essas perguntas se resumem em uma simples questão que o trabalho pretende mostrar: Como é possível unir essas metodologias para criar testes com qualidade e agilidade? Para isso, os próximos tópicos mostra como uni-las e até que ponto as duas metodologias se “aceitam”.

4.2 SCRUM X V-MODEL

Após a explicação breve sobre o SCRUM já se nota algumas diferenças entre essas metodologias. O V-Model é concebido como orientação para o planejamento e execução dos projetos, no qual é nessa metodologia que se define os resultados a serem alcançados. Isto é, a metodologia em cascata é um “portão de decisões” que indica um marco de sequencia, no

qual o estado atual do processo tem que ser avaliado; além de se definir uma quantidade de produtos a serem entregues para “passar” para a outra etapa.

Além disso, o V-Model especifica as responsabilidades de cada participante, ou seja, em termos informais a metodologia descreve em detalhes, "quem" tem a ver 'o que' e 'quando' dentro do um projeto. Já o SCRUM é um quadro simples e baseado em equipes para resolver problemas complexos, no qual suas raízes estão no desenvolvimento de produtos e de encontrar soluções rápidas para qualquer mudança que seja feita dentro do projeto.

Bom, as diferenças entre elas se encontram em simples palavras como enquanto o V-model são processos complexos, o SCRUM deseja a simplicidade. Enquanto uma segue um processo de longo prazo o SCRUM apresenta soluções em curto prazo. Uma a metodologia apresenta equipes grandes e avessas a mudanças a outra com equipes pequenas, mas tranquila caso aconteça de alterações no escopo.

Como em tudo há um pró e um contra nas metodologias não seria diferente. Cada uma possui aspectos interessantes que nesse trabalho será mostrado a visão das qualidades entre elas e não suas diferenças e /ou defeitos. Com isso, deverá ser encontrado o melhor caminho de fazer a aliança entre as duas metodologias “pegando” o que elas possuem de melhor para fornecer para a área de testes.

4.3 ALIANÇA: SCRUM & V-MODEL

A ideia de realizar uma aliança entre as duas metodologias é criar uma forma de beneficiar ainda mais a área de teste tendo como metodologia principal a utilização do V-model utilizando alguns artifícios da metodologia ágil SCRUM para acelerar o processo de entrega dos produtos entre outros benefícios que se obtém conforme a seguir. É importante ressaltar que os processos, técnicas da metodologia V-model não serão tirados, no qual a intenção é adicionar as características do SCRUM para beneficiar ainda mais a área de teste – QA.

Como se percebe a área de teste é a última a atuar em um projeto onde o software é o produto, com isso, caso ocorra algum atraso na entrega do produto a área de teste será a primeira a ser questionada sobre seus “entregáveis” e se estes estão dentro do prazo no cronograma.

Como a metodologia V-model possui as validações e verificações documentadas (*criterias*) é possível verificar qual parte do projeto está atrasada pela equipe de QA e o que já chegou atrasado tendo as documentações como ponto positivo da metodologia em cascata. Entretanto, se utilizar uma das palavras do SCRUM, a simplicidade, com intuito de criar documentações contendo apenas o necessário e não status sem importância já ocorre em grandes melhoras no custo e no tempo do projeto.

Assim criando modelos simplicista para esse tipo de documentação já aumenta a eficácia dos processos visando apenas no que é importante. Além disso, a utilização das reuniões diárias advindas da metodologia SCRUM, caso seja possível na etapa do teste, pode ajudar muito a equipe que está trabalhando em cada processo do V-model entender suas funções e apresentar possíveis problemas que possam estar impactando o projeto. Com isso, será necessário encontrar a melhor união mostrando uma situação em que as duas metodologias se ajudassem.

A ideia é mostrar cada “caixinha” / processo da Figura 6 (capítulo sobre V-model) para exemplificar a associação entre as duas metodologias. Por exemplo, utilizando o processo “Product Test”, ou seja, o teste do produto que contempla a verificação das funcionalidades do sistema. Com isso, para esse processo haverá várias iterações (*sprints*) pensando que cada funcionalidade implementada por desenvolvimento será entregada em datas diferentes.

Seguindo a ideia do SCRUM referente a pessoas haverá um Scrum Master, ou seja, um gerente de projeto que estará “cuidando” de todos os processos do V-model inclusive o Teste de Produto. Além dele, especificamente para esse processo teste de produto, existirá um *Product Owner*, ou seja, o proprietário do produto que irá determinar todos os requisitos que a iteração desse processo irá realizar.

Normalmente, também poderá ser utilizado o nome líder para o proprietário do produto que irá distribuir as atividades para sua equipe sobre a iteração regente. No nosso caso ele irá dividir os cenários de testes das funcionalidades do sistema para sua equipe, no qual terá que

entrega-las em uma data esperada. No final de todas as iterações do *Product Test* será realizada a validação e verificação (*criterias*) para passar para a próxima etapa e equipe.

Um ponto importante é o líder possuir flexibilidade a mudanças, ou seja, caso haja qualquer problema ele deverá avisar o SCRUM Master (gerente de projetos) e juntos estipular novas datas caso afete em todas as iterações do teste do produto, no qual as reuniões referentes aos requisitos poderão ser de grande ajuda em casos como esse.

Finalmente, a uma linha de associação entre as duas metodologias. Isto é, inserindo o SCRUM em cada processo do V-model fazendo com que as duas metodologias sejam utilizadas em conjunto sem tirar suas características e utilizando todos os seus aspectos positivos. Assim podem-se encontrar alguns benefícios:

- A utilização de documentações simplificadas melhora a comunicação entre as equipes e com os clientes.
- A utilização de iterações ajuda no andamento do projeto, pois resumi o que deverá ser testado a cada versão enviada por desenvolvimento.
- As reuniões diárias ou quinzenais ajudam os líderes (product owner) a enxergar a situação da iteração (Sprint) conseguindo solucionar problemas mais rápidos.
- As reuniões realizadas entre lideres e gerentes do projeto facilitam a possibilidade de mudanças críticas exigidas pelo cliente; além do gerente entender todos os problemas que a equipe de determinado líder está passando podendo até ocorrer mudanças de cronograma.

Com isso, pode-se mostrar que há um grande benefício quando as duas metodologias se aliam conseguindo maior agilidade e comunicação entre processos e pessoas em projeto de software.

5 CONCLUSÃO

O trabalhou abordou que a nova área da tecnologia da informação – QA – conseguiu provar seus benefícios mediante os cenários de custo, metodologia, técnica, padrões entre outros mostrando que a qualidade no produto vai além de apenas qualidade, mas sim de muitos outros fatores que empresas inteligentes estão implementando e assim adquirindo lucros e fidelizações de clientes.

A ideia do trabalho é mostrar que a área de teste não é uma “complicadora” na vida das empresas, mas sim uma facilitadora, no qual encontrando os erros é que se nota uma melhora nos produtos, na geração de conhecimento e minimização das insatisfações futuras.

Com isso, há a necessidade de compreender melhor como as metodologias podem influenciar na evolução e eficiência da entrega do produto e que com o auxílio de outras vertentes metodológicas pode-se alavancar ainda mais os processos inovando a maneira das possíveis formas de trabalhar com as metodologias.

Além disso, devesse levar em conta a importância de que há testes específicos para cada particularidade do sistema e com isso a diversos perfis de profissionais que podem trabalhar em uma área de testes. Com testes específicos existem técnicas e ferramentas que devem ser utilizadas de acordo com estes gerando uma grande associação entre todos os processos que envolvem os testes desde a primeira unidade do sistema até a aplicação finalizada para o cliente.

Para tudo isso se manter, as normatizações impostas pelas ISOs entre outro tipos verificam se as qualidades do produto e do processo estão sendo empregadas devidamente. Além de mostrar as melhoras maneiras de empregar a qualidade em uma empresa.

Enfim, o trabalho propôs informar que testes será uma área de grande evolução e disseminação no mercado dos dias atuais. Hoje, dificilmente em uma fábrica de software de grandes empresas não há a existência de testadores no meio de milhares desenvolvedores.

Ainda há muito a evoluir e criar, como mostrar os verdadeiros benefícios de ter a área de QA e que apenas desenvolvedores não são hábeis para realização dos testes onde também há a

desmistificação da guerra entre essas áreas mostrando que uma auxilia a outra nas melhorias dos sistemas.

Finalmente, o trabalho será finalizado, no qual esperasse que a visão dos leitores possa ter mudado referente a área da qualidade de softwares mostrando que não é apenas algo simples de nenhum interesse, mas sim que é uma área de grande valor para o mercado e principalmente para os clientes criando sistemas ágeis, rápidos e com qualidade.

6 BIBLIOGRAFIA

BASTOS, Aderson ... [et al]. Base de Conhecimento em Teste de Software. Traço & Photo editora, 2006.

RIOS, Emerson e Moreira, Trayahu. Teste de Software, 2ª ed. Editora Altabooks, 2006.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C., "Qualidade de software – Teoria e prática". Editora Prentice Hall, 2001.

SCRUM ALLIANCE. Informações sobre Scrum. Disponível em: http://www.scrumalliance.org. Acesso em: 20 de out. 2011.

ACCENTURE BRASIL. ADM - Metodologia de testes. São Paulo, 2010.

Instituto Brasileiro de Qualidade em Testes de Software, Disponível em: http:www.ibqts.com.br. Acessado em: 03 out. 2011.