UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DENISE TOMASI ISOPPO

TESTE DE SOFTWARE BASEADO EM RISCO APLICADO EM NORMAS DE QUALIDADE

CRICIÚMA
2012

DENISE TOMASI ISOPPO

TESTE DE SOFTWARE BASEADO EM RISCO APLICADO EM NORMAS DE QUALIDADE

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador(a): Prof. (ª) MSc. Ana Cláudia Garcia Barbosa

CRICIÚMA
2012

DENISE TOMASI ISOPPO

TESTE DE SOFTWARE BASEADO EM RISCO APLICADO EM NORMAS DE QUALIDADE

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Engenharia de Software.

Criciúma, 26 de Junho de 2012.

BANCA EXAMINADORA

Profª. Ana Claudia Garcia Barbosa- MSc - UNESC - Orientadora

Prof. Paracelso de Oliveira Caldas- MSc - UNESC

Prof. Gustavo Bisognin - MSc - UNESC

Dedico este trabalho a todos que me ajudaram a concluir com sucesso mais esta etapa de minha vida, em especial à minha família que com paciência me ajudou a superar os momentos difíceis. Ao meu namorado que me deu forças para vencer os obstáculos e a Deus que me deu a luz para seguir o caminho dessa jornada.

AGRADECIMENTOS

Agradeço primeiramente a Deus por iluminar meu caminho e me dar forças durante essa caminhada e a toda minha família, principalmente à minha mãe, meu pai, minha irmã, meu namorado, meus avós paternos e maternos, meus tios e tias, que estiveram ao meu lado em todos os momentos difíceis, me apoiando e me dando coragem para seguir em frente.

Agradeço à minha orientadora Ana Cláudia Garcia Barbosa por ter me aceitado como orientanda e pelas horas dedicadas de atenção e paciência durante esse trabalho de conclusão de curso, também ao professor Gustavo Bisognin pela ajuda e atenção prestada.

Ao meu namorado que me apoiou durante esta trajetória, pela compreensão, pelo companheirismo e incentivo concedido. A minha cachorra que agüentou o stress ao meu lado, e aos meus colegas que junto comigo acompanharam esse caminho.

É difícil agradecer todas as pessoas que de algum modo, nos momentos serenos e apreensivos, fizeram ou fazem parte da minha vida, por isso agradeço a todos de coração pelo carinho, incentivo, determinação, apoio, fé e pelo amor de todos.

Muito obrigada por tudo!!

"Construí amigos, enfrentei derrotas, venci obstáculos, bati na porta da vida e disse-lhe: Não tenho medo de vivê-la".
Augusto Cury

RESUMO

Esta pesquisa apresenta como tema central teste de software baseado em risco que consiste num conjunto de atividades que favorecem a identificação de fatores de risco associados ao software, e teve como objetivo a modelagem de um processo de teste de software baseado em risco, o mesmo adequado às normas de qualidade. Esse projeto apresenta um modelo mais amplo do que os encontrados na literatura, pois este consiste em atividades, artefatos, papéis que tem como objetivo guiar os engenheiros de teste na utilização dessa abordagem baseado em risco, de forma a promover melhor uso de recursos e tempo. A composição do processo foi realizada com a utilização da ferramenta Eclipse Process Framework (Epf Composer). Para atender as necessidades do mercado o processo foi adequado as normas de qualidade, utilizando para isso o modelo de processo de software brasileiro (MPS.BR), sendo que o nível de processo abordado foi o nível D.

Palavras chave: Teste de Software, Qualidade de Software, Riscos de Software, Teste de Software Baseado em Risco, Modelo de Processo.

ABSTRACT

This research shows as a main subject software testing based in a risk that consists in a group of activities that favors the identification of risk factors associated to the software, and had as an objective the modeling of a software testing process based in risks, the same appropriated to the quality standards. This project shows a larger modeling than the ones found in the literature, because its consists in activities, artifacts, roles that has as an objective to guide the testing engineers the use in this approach based in risks, as a form to promote a better use from the resources and the time. The composition of the process was done with the utilization from a tool called Eclipse Process Framework (Epf Composer). To meet the needs from the field the process was appropriated to the quality standards using the Brazilian software process modeling (MPS.BR), being the level of the process approached level D.

Word Key: Software Testing, Software Quality, Software Risks, Software Testing based in risks, Modeling Process.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ADR	Análise de Decisão e Resolução
AQU	Aquisição
AMP	Avaliação e Melhoria do Processo Organizacional
APG	Adaptação do Processo Gerência de Projetos
ARC	Análise e Resolução de Causas
BID	Banco Interamericano de Desenvolvimento
CA	Consultores de aquisição
CMM	Modelo de maturidade de capacidade
CMMI	Modelo de maturidade de capacidade integrada
ETM	Equipe Técnica do Modelo
DEP	Desempenho do Processo Organizacional
DRE	Desenvolvimento de Requisitos
FCC	Fórum de Credenciamento e Controle
FINEP	Financiadora de estudos e projetos
GCO	Gerência de Configuração
GQA	Garantia de Qualidade
GQP	Gerência Quantitativa do Projeto
GPR	Gerência de Projetos
GRE	Gerência de Requisitos
GRI	Gerência de Risco
IA	Instituições Avaliadoras
IIO	Inovação e Implantação da Organização
IOGE	Instituição Organizadora de Grupos de Empresas
ISSO	Organização Internacional para Padronização
ITP	Integração do Produto
MA-MPS	Método de avaliação para melhoria do processo de software
MCT	Ministério da Ciência e Tecnologia
MED	Medição
MN-MPS	Modelo de negócio para melhoria do processo de software
MPS.BR	Melhorias de Processo de Software Brasileiro
MR-MPS	Modelo de referência para melhoria do processo de software

PCP	Projeto e Construção do Produto
PMBOK	Base de Projeto de Gestão de Conhecimento
PMI	Instituto de Gerenciamento de Projetos
RBTProcess	Processo de Teste Baseado em Risco
SEI	Instituto de Engenharia de Software
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
TER	Treinamento
VAL	Validação
VER	Verificação
V&V	Validação e Verificação

SUMÁRIO

1 INTRODUÇÃO

Produtos de qualidade, de fácil uso, fácil manutenção são o que todos procuram no mercado, devido a este fato é que hoje a maioria das empresas está aderindo às normas de qualidade.

O mercado de software está muito concorrido, com isso as empresas estão em busca constante de melhorias, cada qual buscando se adequar às normas de qualidade existentes. Nos últimos anos, as empresas de software no Brasil favoreceram a ISO 9000 em detrimento de outras normas e modelos especificamente voltadas para a melhoria de processos de software (MIT, 2003).

Com a necessidade de um aprimoramento e um modelo mais adequado para as empresas, surgiu o modelo brasileiro de qualidade, o manual de Melhoria de Processos do software Brasileiro (MPS.BR) baseado no CMMI, nas normas ISO/IEC 12207 e ISO/IEC 15504 e na realidade do mercado nacional, é simultaneamente um movimento para a melhoria e um modelo de qualidade de processo voltada para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil.

O desenvolvimento de produtos de software está em constante evolução, e as empresas desenvolvedoras vêm enfrentando problemas quanto ao usuário final do software, a cada dia aumenta o nível de exigência por parte dos clientes no que diz respeito ao controle de qualidade do produto, no desempenho, complexidade e nos efeitos que o seu uso possa produzir na organização.

As falhas de sistema que chegam ao usuário, por exemplo, são fatores comuns nas companhias e os mais preocupantes. Apesar dos testes serem atividades difíceis e de alto custo para ser realizado em uma empresa devido aos inúmeros erros possíveis, os mesmos se tornaram parte essencial no desenvolvimento de um software. Segundo Kaner (1999), e demais autores, a atividade de teste de software é bastante cara, chegando a custar até 45% do valor inicial de um produto em desenvolvimento.

Além das possíveis falhas, os eventuais problemas afetados por riscos imprevistos que ocorrem comumente em projetos, aqueles não planejados ou simplesmente os ignorados, sendo que com o aumento da complexidade e o

crescimento do projeto os mesmos se agravam cada vez mais, tornando cada vez piores de serem resolvidos.

Se a área de teste não estiver bem organizada os defeitos vão persistir a ocorrer num estágio onde os custos são cada vez maiores.

Com intuito de minimizar problemas como estes as empresas estão buscando se aprimorar mais nas atividades teste de software, sendo que os mesmos passaram a ser executados por equipes especializadas e as empresas criaram áreas na sua estrutura organizacional para poder cumprir esse papel.

O teste de software baseado em risco é aplicado no planejamento dos testes associados aos requisitos do software. Conhecido pelo termo em inglês risk – based testing, consiste em identificar, priorizar e criar casos de teste que exploram os riscos no sistema, e tem se mostrado de grande utilidade para as empresas, aproveitando melhor os recursos e o tempo do teste, perfazendo um processo que aborde as áreas de maior risco do sistema e garantindo que o produto final não saía errado.

Com base neste contexto, esta pesquisa propõe definir um processo de caso de teste baseado em risco, visando a aplicação das normas de qualidade MPS.BR.

1.1 OBJETIVO GERAL

Realizar a Modelagem de Processo de Teste de Software Baseado em Risco Aplicado na Norma de Qualidade MPS.BR

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do projeto são baseados nos seguintes itens:

a) compreender o modelo de referência MPS.BR,
b) entender o nível D de maturidade do modelo MPS.BR,
c) estudar os modelos de processos de teste de software,
d) demonstrar processos de testes baseados em risco,
e) modelar processo de teste dentro do nível D de maturidade do modelo de qualidade MPS.BR,

1.3 JUSTIFICATIVA

Segundo Rios (2007. p. 93), “o risco é um dos elementos mais importantes a serem analisados e trabalhados no momento da elaboração de um projeto de teste de uma aplicação”.

Com a evolução das tecnologias os sistemas ficaram mais complexos, dificultando a execução dos testes dentro do modelo tradicional, estes que eram testados dentro do processo de desenvolvimento. Os programadores e analistas de sistemas não são testadores e não são especializados na execução deste tipo de atividade. Os programas, por vezes, são liberados para os clientes com defeitos, os quais precisam ser corrigidos acarretando custos cada vez maiores. Uma possível solução a esta questão pode ser a adesão de um processo separado do processo de desenvolvimento, onde essa atividade seria exercida por técnicos treinados e qualificados para executar tal função.

Dentre as técnicas de verificação e validação de softwares, a atividade de teste é uma das mais utilizadas para avaliar a qualidade do software com finalidade de minimizar a ocorrência de erros e riscos associados. Sendo que o teste baseado em risco está sendo uma ótima opção para as empresas do ramo.

O foco do teste baseado em risco é na análise do software e na criação de um plano teste baseado nas áreas que possuam a maior probabilidade de apresentarem problemas e que teriam o maior impacto sobre o mesmo (MCMAHOM,1998 apud OSENBERG; STAPKO; GALLO, 1999).

A atividade de teste de software anda paralelamente ao risco. Ao avaliar os riscos de um projeto, as equipes de análises de teste buscam aqueles fatos cujas ocorrências poderão acarretar em perdas para a empresa. Quando um testador por motivo de tempo ou por qualquer outro motivo, deixa de testar uma parte do sistema, ele está assumindo um grande risco já que um componente do sistema não será testado. Se esse risco vier a se materializar, ou seja, essa parte do sistema que não foi testada conter um erro, a empresa poderá sofrer uma grave perda financeira e até de confiança por parte do usuário final.

De maneira mais ampla o processo de teste de software baseado em riscos seria descrito de tal maneira; processo de identificar os riscos de um sistema, analisar os mesmos, priorizá-los, criar casos de teste que mitiga os riscos identificados e gerenciar sua execução.

Amland (1999), foca suas metodologias na utilização de riscos para fins de teste, explicitamente com a finalidade de reduzir o custo na fase de teste do projeto e a redução de futuros custos de produção em potencial pela otimização do processo de teste.

Sua maior vantagem sem dúvida está relacionada ao ganho de custo e prazo no processo de teste, incluindo outras tais como agilidades nos processos de teste, proteção a vítimas/cliente de possíveis erros no sistema, fazer melhor uso dos recursos de teste.

Com isso, pode-se observar que as empresas estão buscando, cada vez mais, a maturidade nos seus processos de software, equipes qualificadas na questão teste para atingir níveis de qualidade e produtividade sempre maiores, que são essenciais para a sobrevivência no mercado de tecnologia da informação, tendo em vista que não adianta apenas testar, mas sim testar bem. Por isso, quando se inicia um projeto de desenvolvimento de software, deve ser também iniciado um projeto de teste de software.

Por fim, este trabalho se justifica pelo fato de mostrar não só que é essencial determinar processos de teste baseados em risco antes da implementação de um software, mas proporcionar às empresas a possibilidade de adotar processos de testes a serem tomados como base na hora do desenvolvimento do software para que não ocorram riscos inesperados e se tornem problemas maiores no futuro.

1.4 ESTRUTURA DO TRABALHO

A estrutura da presente pesquisa é dividida em 8 capítulos, sendo que no primeiro capitulo é apresentado a introdução, os objetivos gerais e específicos, e a justificativa do tema proposto.

O segundo capítulo apresenta conceitos para que se construa um software de qualidade e algumas normas e modelos como, ISO 15504 e ISO 12207, CMMI e o modelo brasileiro MPS.BR, que fazem parte para que o mesmo seja desenvolvido.

O capítulo 3 apresenta como assunto, os testes de softwares, análise de risco, tipos de risco que podem acontecer, o que fazer para evitá-los de forma a avaliar ou melhorar o software com as técnicas de teste apresentadas, validação e verificação.

No capítulo 4 serão apresentados conceitos do processo de teste de software, ciclo de vida dos processos de teste, e alguns modelos de teste. Os processos de teste de software e suas particularidades serão apresentados no capítulo seguinte.

No sexto capítulo será apresentado como assunto, teste de software baseado em riscos, e suas definições.

Os trabalhos correlatos serão apresentados no capítulo 7, e por fim o projeto de pesquisa será apresentado no capítulo 8. No último capítulo serão apresentados a modelagem de processo de teste baseado em risco e também a aplicação das normas de qualidade que será aplicada no processo para fazer a validação do mesmo.

2 QUALIDADE DE SOFTWARE

Qualidade de acordo com o dicionário Aurélio se define como um atributo, condição, uma propriedade das pessoas ou coisas, onde se possa distingui-las de outras podendo determinar a natureza das mesmas.

A busca da qualidade hoje, é grande preocupação das organizações, pois desenvolver um software altamente qualificado não é mais um aperfeiçoamento para poucos, e sim um fator de grande competitividade no mercado, este cada vez mais exigente.

De acordo com Koscianski e Soares (2007), qualidade não é um termo trivial e a busca deste conceito não é fácil, é um tanto subjetivo. A idéia de qualidade é aparentemente indutiva, entretanto quando examinada detalhadamente o conceito se define complexo.

Pressman (2007) define qualidade em software de forma sendo que antecipe e satisfaça os requisitos dos clientes, e escreva tudo o que se deva fazer e fazer tudo o que foi escrito estando sempre em conformidade com requisitos.

Muitos fatores levam os clientes e as próprias empresas a buscarem cada vez mais a qualidade de seus softwares, como a complexidade das aplicações, o surgimento da internet deixando a imagem das empresas exposta a um grande público, e por fim a crescente concorrência no mercado de desenvolvimento (RIOS, 2007).

A qualidade do software tornou-se requisito indispensável para que um software permaneça no mercado, sendo que as empresas mais competitivas procuram trabalhar sempre em busca da melhoria contínua dos processos para aumentar a qualidade do processo de desenvolvimento, assim aumentando a qualidade do produto final.

Um dos maiores problemas encontrados na engenharia de software é saber se um software é ou não de qualidade, e como fazer para defini-lo.

O que se espera de um software de qualidade profissionalmente desenvolvido pode-se dizer, segundo Pressman (2008), que é a conformidade a requisitos funcionais e de desempenho declarados nitidamente, a padrões de desenvolvimento explicitamente documentados e a características implícitas.

Segundo Gomes (2000), de acordo com a norma internacional ISO/IEC 9126, onde define qualidade de software como um conjunto de características que

consta em um produto de software que lhe confere a capacidade de satisfazer necessidades implícitas e explícitas.

As necessidades implícitas são necessidades particulares dos usuários, os chamados fatores externos e que podem ser percebidas tanto pelos desenvolvedores como pelos usuários, como também podem ser chamadas de qualidade em uso, que devem permitir aos usuários que atinjam metas com produtividade, satisfação, efetividade e segurança em um contexto de uso especificado.

Por sua vez as necessidades explícitas se dizem respeito aos objetivos e condições propostos pelos desenvolvedores dos softwares. Estes são fatores relativos à qualidade do processo de desenvolvimento do produto e são vistos apenas pelos que produzem.

Sendo assim percebe-se que qualidade de software não é uma idéia tão simples, é mais fácil descreve-la como um conjunto de atributos ou fatores especificados que variam de acordo com as diferentes aplicações e os clientes que as solicitam, já a garantia de qualidade de um software é aplicada ao longo de todo processo de engenharia de software, e consiste de técnicas, procedimentos e ferramentas aplicadas por profissionais para assegurar que um produto excede ou atinge padrões predefinidos durante o ciclo de desenvolvimento do produto (BUENO; CAMPELO, 2011).

Qualidade pode ser definida de várias formas, dependendo do ponto de vista sob qual é analisado, e na questão qualidade de software pode ser resumida como uma série de atividades que deve ser seguida para que seja garantido ao cliente que o produto final seja construído de forma eficiente e que atenda as expectativas esperadas pelo mesmo, sendo que metodologias, normas e padrões corretos tenham sido usadas para sua implementação.

2.1 NORMAS DE QUALIDADE

Devido ao crescimento do desenvolvimento de softwares por pequenas e médias empresas, essas precisam suprir a necessidade do rigoroso controle de qualidade exigida pelos consumidores.

Estudos comprovam que a grande maioria dos softwares existentes no mercado não atende aos objetivos projetados, devido à falta de processos

adequados nas organizações em que são desenvolvidos. A partir daí a necessidade fez com que tivesse uma constante evolução no desenvolvimento de modelos de maturidade dos processos de software em todo mundo.

Surgiu assim, através do SEI – Software Engineering Institute (SEI, 2009a) que publicou o SW-CMM, Capability Maturity Model for Software em 1987 e o CMMI, Capability Maturity Model Integration em 2002 e da ISO/IEC – International Organization for Standardization / International Electrotechnical Commission (ISO, 2009) também publicou as normas ISO/IEC 15504 e ISO/IEC 12207 nos anos 90.

Os padrões e normativas servem para medir os aspectos da qualidade de software, como a qualidade do produto de software, qualidade do processo de desenvolvimento e o nível de maturidade que a empresa que adapta, desenvolve, customiza o software.

O processo de desenvolvimento e manutenção de um software é diferente de qualquer outro produto industrial, sendo que nesse campo tecnológico que se desenvolve rapidamente sejam necessárias orientações adicionais para o estabelecimento de sistemas de qualidade onde estejam envolvidos os produtos de software.

Segundo Abnt (2000), com o considerável crescimento das indústrias de software e levando em conta que a criação de software apresenta peculiaridades, a ISO que há muito tempo relaciona-se à qualidade, tem trabalhado no esclarecimento de inúmeras normas para que possam ser utilizadas como padrões e guias para várias áreas de atuação dentro do contexto do mercado de software. A ISO foi criada para suprir a carência de referências internacionais para regulamentação das obrigações pactuadas entre compradores e fornecedores, e manter a garantia da uniformidade da qualidade de produtos e manutenção.

A definição da ISO, controle de qualidade, é a atividade e técnica operacional que utilizada serve para satisfazer os requisitos de qualidade.

Segundo Ferreira (2009), o Brasil é um país cujo desenvolvimento de software só vem crescendo nos últimos anos e está entre um dos maiores desenvolvedores de software no mundo. Para se obter uma certificação com intuito de atingir a produtividade e qualidade internacional, o custo do mesmo se torna elevadíssimo tornando-se inviável para pequenas e médias empresas. Sendo assim a Softex, Governo e Universidades resolveram formar uma parceria chamada MPS.BR, sendo uma solução baseada na realidade brasileira.

Em 2005, para melhoria dos processos de software foi publicada e primeira versão do Guia Geral do programa para Melhoria de Processo do Software Brasileiro (MPS.BR), possibilitando às empresas atingir maturidade em seus processos.

Segundo o guia da Softex diz que o MPS.BR é adequado ao perfil de empresas de diferentes características e tamanhos.

> Busca-se que este modelo seja adequado ao perfil de empresas com diferentes tamanhos e características, públicas e privadas, embora com especial atenção às micros, pequenas e médias empresas. Também se espera que o modelo MPS seja compatível com os padrões de qualidade aceitos internacionalmente e que tenha como pressuposto o aproveitamento de toda a competência existente nos padrões e modelos de melhoria de processo já disponíveis (SOFTEX, 2009a p. 6).

Qualidade ou melhoria de processo do software está associada ao CMMI, sendo que algumas normas como ISO/IEC 12207 e ISO/IEC 15504 foram as bases para criação desses modelos, CMMI e o MPS.BR.

Os modelos ISO 9001, 9126 e 12207, e as normas ficaram conhecidos por serem voltados à qualidade do produto, uma vez que estava mais perto da percepção do cliente, este que à alguns tempos atrás se preocupava apenas em ter um software sem falhas.

Segundo Côrtes e Chiossi (2001), com a evolução tecnológica das últimas décadas, entre outros fatores, elevou consideravelmente a expectativa dos usuários. Por esse motivo, a inexistência de bugs, passou a ser um requisito básico acompanhado de características como portabilidade, usabilidade, entre outros, deixando de ser um diferencial competitivo. Desta forma as empresas estão procurando se adequar às novas demandas que o mercado lhe impôs.

2.1.1 ISO 12207 e ISO 15504

De acordo com a Softex (2009a), a norma ISO 12207 foi criada em conjunto pela ISO, International Organization for Standardization, e o IEC, International Electrotechnical Commission. Foi publicada como norma internacional em 1995 e só passou a ter sua versão brasileira em 1998 com o mesmo nome, acrescendo somente o prefixo NBR.

A norma ISO/IEC 12207 está relacionada aos processos de ciclo de vida de software e tem como objetivo principal, o estabelecimento de uma estrutura comum para os mesmos, como forma de ajudar as organizações a compreenderem todos os componentes presentes na aquisição e fornecimento de software fazendo com que consigam firmar contratos e executem projetos de forma mais eficaz (MACHADO, 2006). Essa norma contém atividades, processos e tarefas para poder serem aplicados durante a obtenção, desenvolvimento, abastecimento e preservação de produtos de software e serviços futuros.

De acordo com Ferreira (2009), em outubro de 2002 e 2004 surgiram atualizações e foram incorporadas algumas melhorias, onde passou a ser chamada emenda 1 e 2 respectivamente, fazendo com essas modificações representassem a evolução da Engenharia de Software, as necessidades vivenciadas pelos usuários da norma e o ajustamento com a série ISO/IEC 15504, avaliação de processo. Em 2008, passou por uma reformulação completa em sua estrutura.

Esta norma também contém processos definidos que devem ser utilizados como referência na implementação do MR- MPS e na avaliação acompanhando o MA-MPS. Com essa norma é possível a realização de inclusões, alterações e exclusões de processos que não sejam relativos ao negócio, seguindo o processo de adaptação da NBR ISO/IEC 12207 (FERREIRA, 2009).

Por sua vez a norma NBR ISO/IEC 15504, surgiu de um estudo realizado em setembro de 1992, chamado “Necessidades e Exigências para uma Norma de Avaliação de Processos de Software”, onde se concluiu que era necessária a elaboração de uma norma que fosse empregável à melhoria de processos e à prescrição da capacidade. A mesma define um modelo bidimensional, que tem como objetivo a realização de avaliações sistemáticas de processos de software como foco na melhoria dos processos, assim identificando os pontos fracos e fortes que serão utilizados para a elaboração de um plano de melhorias, gerando um perfil dos processos e a determinação da capacidade dos processos viabilizando a avaliação de um fornecedor em potencial e a determinação do poder de processos de uma organização.

Esta define também um guia para a orientação da melhoria de processos tendo como referência um modelo de processo e como uma das etapas a realização de uma avaliação de processos (ABNT NBR ISO 15504-2, 2008).

Segundo Côrtes (2001), a organização deve definir o contexto e os objetivos, também escolher o modelo e o método para a avaliação e definir os objetivos de melhoria.

2.1.2 CMMI

O modelo SW-CMM de qualidade de software tornou-se um modelo de qualidade respeitado pelas empresas de engenharia de software por ser eficiente e eficaz.

O modelo CMM, foi desenvolvido pelo SEI, consiste em uma organização ligada à universidade Carnegie Mellon. O desenvolvimento do mesmo foi financiado pelo Departamento de Defesa Americano, com a finalidade de estabelecer um padrão de qualidade para software desenvolvido para as forças armadas (PÊSSOA, 2005).

Segundo Mertins (2004), seus trabalhos tinham como principal objetivo, adaptar uma metodologia de qualidade de processos para a área de software, que pudesse ser adotada por órgãos do governo americano, assim estabelecendo um nível de maturidade que um prestador de serviços de software deveria ter para atender suas demandas. Sendo assim, de acordo com Palza, Fuhrman e Abran (2003), o CMM se tornou o principal modelo de referência para avaliação de maturidade de processos de software em empresas de desenvolvimento de software e ficou conhecido internacionalmente.

O CMM descreve elementos importantes da evolução de um processo de software imaturo para um processo maduro e disciplinado, abrangendo práticas para planejamento, engenharia e gestão do desenvolvimento de software que, quando seguidas, melhoram a habilidade da organização em atender metas de prazos, custos, funcionalidades e qualidade do produto final.

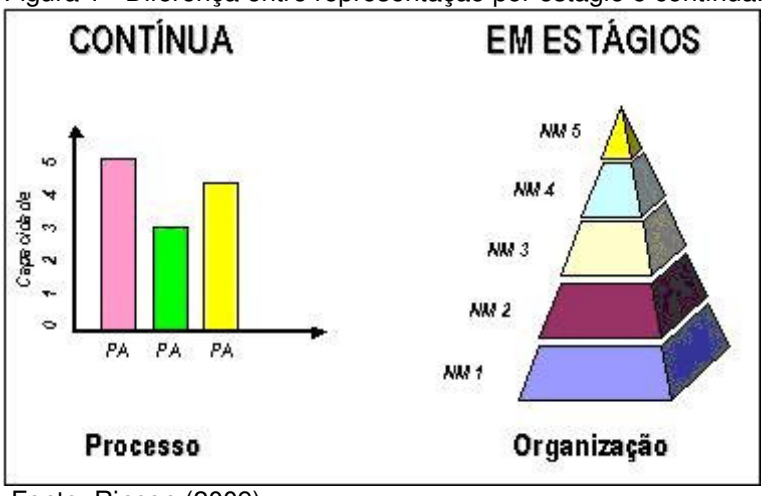
A partir de 1991 foram desenvolvidos CMMs® para várias áreas, mas apesar de ser de grande utilidade, o uso de múltiplos modelos apresentou diversos problemas, visando assim a necessidade de um novo modelo.

Em 2002 surgiu o modelo integrado de maturidade da capacidade, CMMI, a incrementação da letra I, integrado, ao CMM se dá pelo motivo de integrar os modelos antigos do CMM em uma estrutura alinhada e coerente. A maturidade da capacidade está relacionada ao processo de desenvolvimento do software.

Segundo Sei (2006), o CMMI foi elaborado para verificar a maturidade atual do processo e reconhecer as questões mais críticas para a qualidade e melhoria do processo de software, onde o nível de maturidade da unidade organizacional exprimiu graus de melhoria na execução dos processos.

Com singelos ajustes e melhorias, em 2006, foi lançada a versão 1.2 do CMMI, para que este fosse compatível com a norma ISO 15504, sendo sua representação de forma contínua, onde possibilita que uma organização tenha sua capacidade avaliada por processos e que cada um tenha níveis diferentes de capacidade, já a versão antiga era representada por estágios, na qual define um conjunto de áreas de processo para definir uma trilha de melhoria para unidade organizacional seguir, esboçando em termos de níveis de maturidade. (RINCON, 2009). A diferença entre os modelos segue na figura 1.

Figura 1 - Diferença entre representação por estágio e contínua.



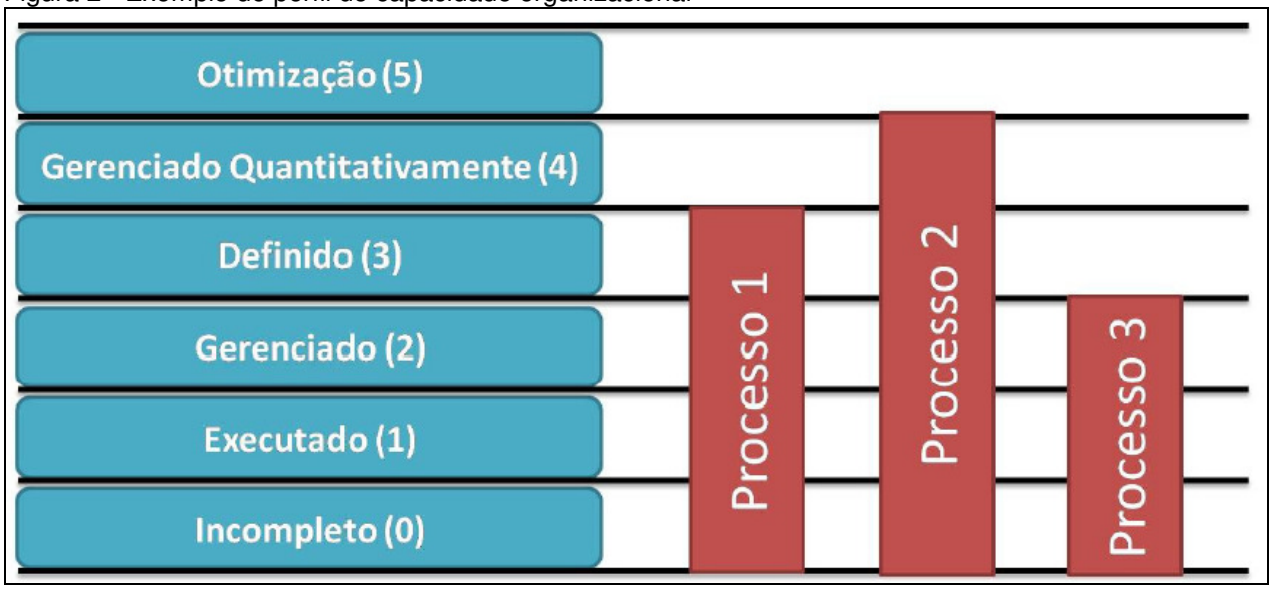
Fonte: Rincon (2009)

O CMMI por estágios não demonstra mudanças relacionado ao CMM. Ou seja, a ideia é ir implementando os processos conforme a seqüência dada pelo modelo CMMI-DEV, CMMI de desenvolvimento (SEI, 2009b). Já a representação contínua possibilita que uma organização tenha sua capacidade avaliada por processos e que cada um tenha níveis diferentes de capacidade.

De acordo com Pêssoa (2005), para poder compreender o significado do termo capacidade deve-se comparar com a capacidade que uma pessoa possui para realizar uma tarefa, sendo que quanto maior a capacidade do processo, melhor deve ser o resultado a ser obtido. Quando uma organização é avaliada levando-se

em conta a capacidade de seu processo, o resultado será seu perfil de capacidade, onde cada processo se encontra em um nível diferente, representado na figura 2.

Figura 2 - Exemplo de perfil de capacidade organizacional



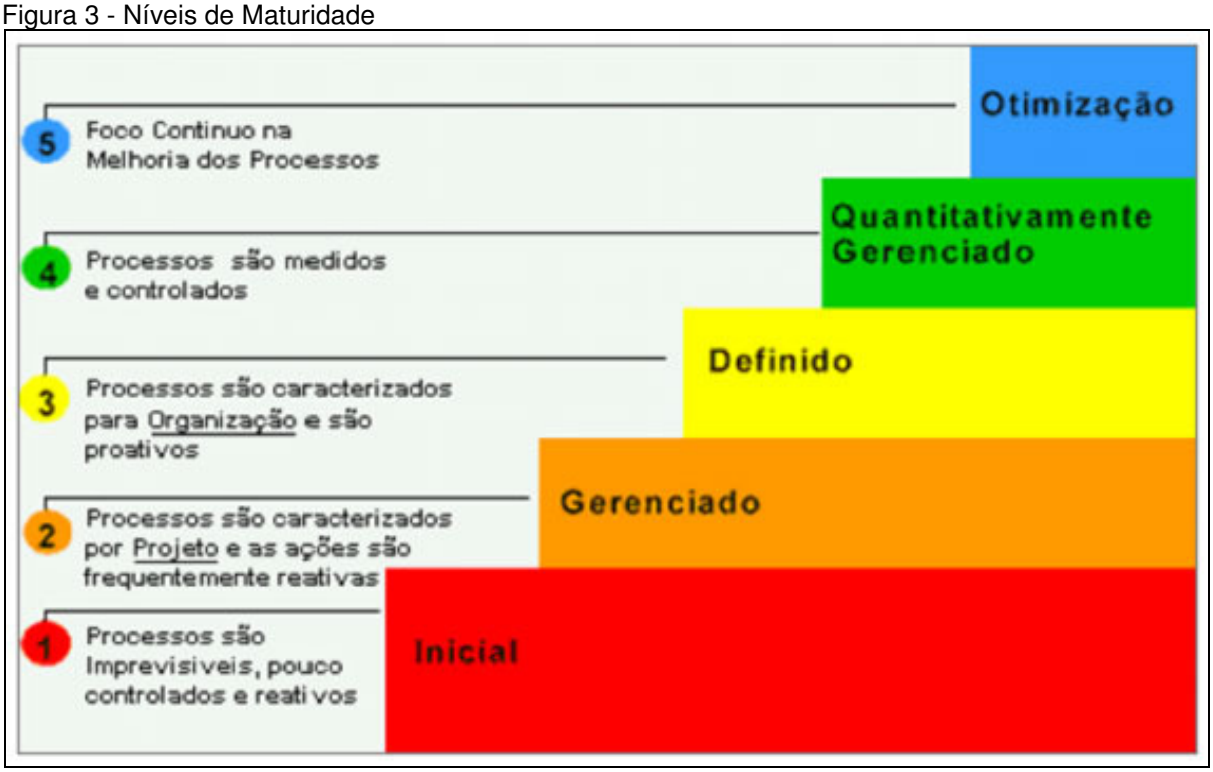
Fonte: Pêssoa (2005)

Sub-seguinte será apresentado seus níveis, no modelo contínuo:

a) **nível 0:** incompleto: o processo não é executado ou é executado parcialmente;
b) **nível 1:** executado: o processo satisfaz as metas específicas da área de processo;
c) **nível 2:** gerenciado: o processo é executado e também monitorado, planejado e controlado para atingir um objetivo;
d) **nível 3:** definido: o processo é gerenciado, adaptado de um conjunto de processos padrão da organização;
e) **nível 4:** gerenciado quantitativamente: o processo é definido, controlado utilizando estatística ou outras técnicas quantitativas;
f) **nível 5:** otimização: o processo é gerenciado quantitativamente para a melhoria contínua do desempenho do processo.

Segundo Rincon (2009), no modelo por estágios uma organização deve implementar os processos de acordo com uma seqüência pré estabelecida, sendo diferente da representação contínua onde a organização é avaliada por um conjunto de processos de acordo com o nível que se constatar. Os níveis são denominados níveis de maturidade e variam de 1 a 5 conforme segue na figura 3.

Figura 3 - Níveis de Maturidade



Fonte: Rincon (2009)

A descrição dos seus níveis segue:

a) **nível 1 (Inicial):** nesse nível o processo é caracterizado como sendo imprevisível, poucos processos são definidos e o sucesso depende de esforços individuais. Algumas características desse nível são: processos que são fracamente controlados, processos que não são passíveis de repetição, há pouca, ou nenhuma, padronização das atividades de gerenciamento e desenvolvimento, o processo de software é altamente dependente das pessoas que nele atuam;

b) **nível 2 (Gerenciado):** processos básicos de gerenciamento de projeto são estabelecidos para controle de custos, prazos e escopo. O controle de processo permite repetir sucessos de projetos anteriores em aplicações futuras e semelhantes. É caracterizado por: Introdução de gerenciamento de requisitos dos projetos, os processos são planejados, passíveis de medição e controle, o processo de software torna-se passível de repetição, os processos, os produtos de trabalho e os serviços são gerenciados;

c) **nível 3 (Definido):** um processo formado por atividades de gerenciamento, é documentado, padronizado e integrado em um

processo padrão da organização. Todos os projetos utilizam uma versão aprovada e adaptada do processo organizacional para desenvolvimento e manutenção de produtos e serviços tecnológicos. As características são: Abrange todas as áreas de processo do nível gerenciado, evolui no sentido de que o processo é descrito em padrões, procedimentos bem definidos, ferramentas e métodos;

d) **nível 4 (Gerenciado Quantitativamente):** são coletadas métricas detalhadas dos processos e projetos, que neste momento são compreendidos e controlados de forma quantitativa. Caracteriza-se por: Todas as metas dos níveis gerenciado e definido serem atingidas, o processo evolui com a adoção de técnicas estatísticas, medidas de desempenho do processo são coletadas e analisadas quantitativamente;
e) **nível 5 (Em Otimização):** a melhoria contínua do processo é estabelecida por meio de sua avaliação quantitativa, e da implantação planejada e controlada de tecnologias e idéias inovadoras. Esse nível é caracterizado por: todas as áreas de processos dos níveis gerenciados, definidos e gerenciados quantitativamente são atingidas, o processo é melhorado continuamente baseado em um entendimento quantitativo das causas de variação comuns inerentes aos processos de desenvolvimento.

Conforme ilustrado na figura 3, o CMMI, no modelo por estágios, é composto por 5 níveis de maturidade. Pode-se considerar, segundo o CMMI, que toda empresa que ainda não passou por uma implementação de melhoria de processos, está no nível inicial, sendo assim, para se tornar mais madura em relação ao seu processo de desenvolvimento, deve-se primeiramente implementar os processos do segundo nível e solicitar uma avaliação por uma instituição credenciada. Ela deverá realizar esse procedimento até que alcance o último nível, 5, que é o mais alto na escala do CMMI.

Cada um desses níveis está dividido em áreas de processo, e cada área de processo possui dentro do documento que descreve o modelo CMMI-DEV.

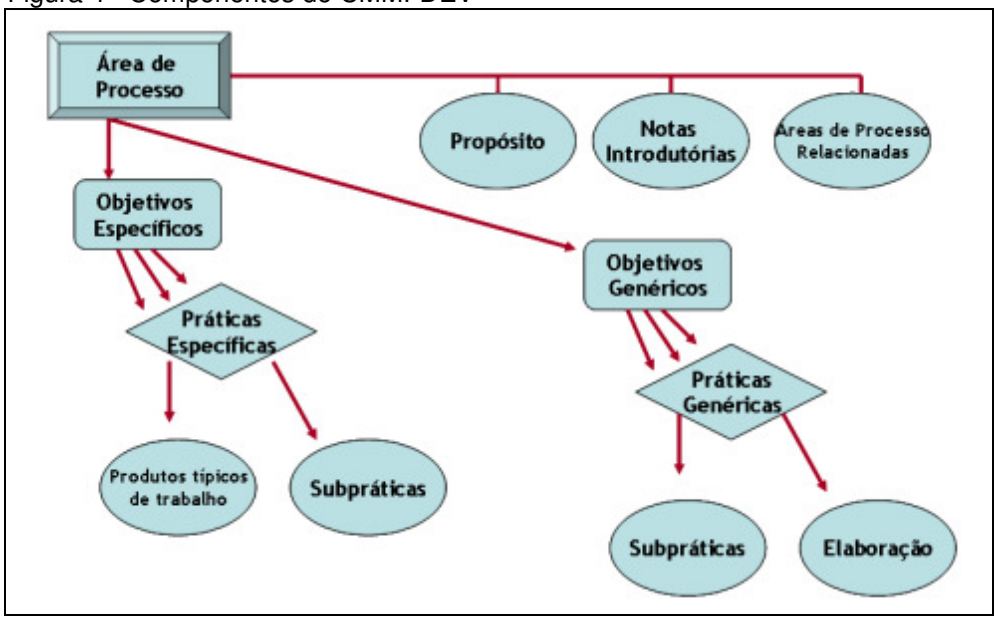
Diversos outros CMM's foram criados, procurando cobrir outras áreas de interesse. Assim, surgiram modelos (Belloquim, 2002): SA-CMM, SE-CMM, IPDCMM, P-CMM.

O CMMI-DEV, voltado ao processo de desenvolvimento de produtos e serviços, é o modelo mais recente de maturidade para desenvolvimento de software do SEI. Além disso, o novo modelo reforça aspectos relacionados à gestão de fornecedores e poderá assimilar outros processos futuramente, o foco do CMMI-DEV é atuar no desenvolvimento e não na operação.

A maturidade de um processo reflete na medida ele pode ser definido, gerenciado, medido, controlado e executado de maneira eficaz.

Em relação à sua estrutura, o CMMI-DEV é formado por componentes agrupados em três categorias: componentes requeridos, componentes esperados e componentes informativos, que auxiliam na interpretação do modelo, conforme mostra a figura 4.

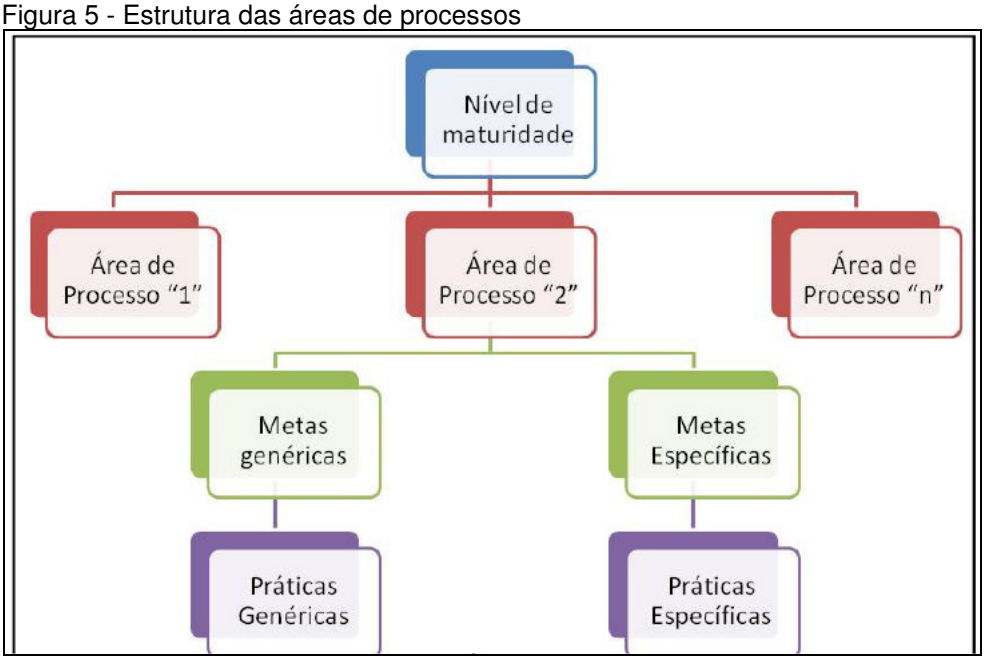
Figura 4 - Componentes do CMMI-DEV



Fonte: Sei (2006)

Cada área de processo é um conjunto de práticas relacionadas que, juntamente implementadas, satisfazem os objetivos considerados importantes para constituir a melhoria do processo e consequentemente da empresa. O CMMI-DEV de acordo com Sei (2006), é composto por 22 áreas de processos e suas metas e práticas são bem especificadas e detalhadas como podem ser observadas na figura 5, com os respectivos níveis de maturidade e categorias representados no quadro 1.

Figura 5 - Estrutura das áreas de processos



Fonte: Sei, (2006a)

Quadro 1- Áreas de Processo do CMMI-DEV

Nível de maturidade	Área de Processo	Categoria
2	Monitoração e Controle do Projeto (PMC)	Gerência de Projeto
2	Planejamento do Projeto (PP)	Gerência de Projeto
2	Gerência de Requisitos (REQM)	Engenharia
2	Análise e Medição (MA)	Apoio
2	Garantia da Qualidade do Processo e do Produto (PPQA)	Apoio
2	Gerência de Configuração (CM)	Apoio
3	Gerência de Fornecedor Integrada (SAM)	Gerência de Projeto
3	Gerência de Projeto Integrada (IPM)	Gerência de Projeto
3	Gerência de Riscos (RSKM)	Gerência de Projeto
3	Definição do Processo Organizacional (OPD)	Gerência de Processo
3	Foco no Processo Organizacional (OPF)	Gerência de Processo
3	Treinamento Organizacional (OT)	Gerência de Processo
3	Desenvolvimento de Requisitos (RD)	Engenharia
3	Integração do Produto (PI)	Engenharia
3	Solução Técnica (TS)	Engenharia
3	Validação (VAL)	Engenharia
3	Verificação (VER)	Engenharia
3	Análise de Decisão e Resolução (DAR)	Apoio
4	Gerência Quantitativa do Projeto (QPM)	Gerência de Projeto
4	Desempenho do Processo Organizacional (OPP)	Gerência de Processo
5	Inovação Organizacional e Posicionamento Estratégico (OID)	Gerência de Processo
5	Resolução e Análise Causal (CAR)	Apoio

Fonte: Sei, (2006)

O propósito, os objetivos específicos, aqueles relacionados ao processo, e os objetivos genéricos, relacionados a todos os processos e à organização, compõem as áreas de processo. Para que uma determinada área de processo seja atendida, têm que serem definidas as características únicas, esta que é a função

dos objetivos específicos que possuem um conjunto de práticas específicas que são as descrições de atividades consideradas importantes para que seja satisfeito.

Os objetivos genéricos, por sua vez, estão associados a mais de uma área de processo e definem as características que devem estar presentes para institucionalizar os processos que implementam a área de processo. Uma prática genérica é a descrição de uma atividade dita importante para a satisfação de um objetivo genérico.

Em processos de avaliação da implementação do CMMI-DEV a equipe de avaliação visa buscar evidências de que as práticas e objetivos, específicos e genéricos, sejam atendidos nos projetos, para as respectivas áreas de processos avaliadas (SEI, 2006).

2.1.3 MPS.BR

Devido à falta de um processo de desenvolvimento de software a maioria das empresas sempre se deparava com problemas como, projetos em produção sem controle de escopo, atrasos no cronograma, os requisitos não eram registrados e nem as mudanças de requisitos eram gerenciadas, falta de definições nos contratos com clientes, entre outros.

De acordo com dados levantados pelo do Ministério da Ciência e Tecnologia (MCT), mais de duzentas empresas de desenvolvimento de software no Brasil possuíam certificação ISO 9000, enquanto menos de cinquenta possuíam certificado SEI/CMU de avaliações CMM®. Destas últimas, a maioria subsidiária de grandes empresas multinacionais, sendo que a maioria está no nível 2 e apenas uma empresa se encontra no nível 5 (MCT, 2008).

Para ajudar na solução destes problemas, no dia 11 de dezembro de 2003, em uma reunião realizada pelo MCT em Brasília, a SOFTEX, Associação para Promoção da Excelência do Software Brasileiro, lançou o Programa MPS.BR (SOFTEX, 2007).

O MPS.BR é um programa para Melhoria de Processo do Software Brasileiro, é simultaneamente um movimento para a melhoria e um modelo de qualidade de processo voltada para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil, coordenado pela SOFTEX, contando com apoio do Ministério da Ciência e Tecnologia (MCT), da Financiadora

de Estudos e Projetos (FINEP) e do Banco Interamericano de Desenvolvimento (BID) (MPS.BR, 2007a).

A coordenação do programa MPS.BR conta com o apoio de duas estruturas para o desenvolvimento de suas atividades, sendo uma delas o Fórum de Credenciamento e Controle (FCC) e outra a Equipe Técnica do Modelo (ETM) (MPS.BR, 2007b). É por meio destas estruturas, o MPS.BR obtém a participação de representantes de instituições governamentais, centros de pesquisas, universidades, e de organizações privadas, os quais contribuem com suas visões complementares que agregam qualidade ao empreendimento.

O Modelo MPS.BR é baseado nas normas ISO/IEC 12207, ISO/IEC 15504 e no CMMI, e na realidade do mercado brasileiro. No Brasil, uma das principais vantagens do modelo é seu custo reduzido de certificação em relação às normas estrangeiras.

Um dos objetivos do projeto é replicar o modelo na América Latina, incluindo o Chile, Argentina, Costa Rica, Peru e Uruguai.

SOFTEX (2009a), afirma que o MPS.BR é baseado em duas metas:

a) **técnicas:** visa à criação e ao aprimoramento do modelo MPS esperando os seguintes resultados: guias do modelo MPS; Instituições Implementadoras credenciadas para prestar serviços de avaliação seguindo o método de avaliação MA-MPS; consultores de aquisição (CA), certificados para prestar serviços de consultoria de aquisição de software; e serviços relacionados;

b) **de Mercado:** visa à disseminação e à adoção do modelo MPS em todas as regiões do país, num intervalo de tempo justo, a um custo razoável. Sendo esse custo tanto em PME (foco principal) quanto em grandes organizações públicas e privadas, esperando os seguintes resultados: criação e aprimoramento do modelo de negócio MN-MPS; cursos, provas e workshops; organizações que implementaram o modelo MPS; organizações com a avaliação MPS publicada tem um prazo de validade de três anos.

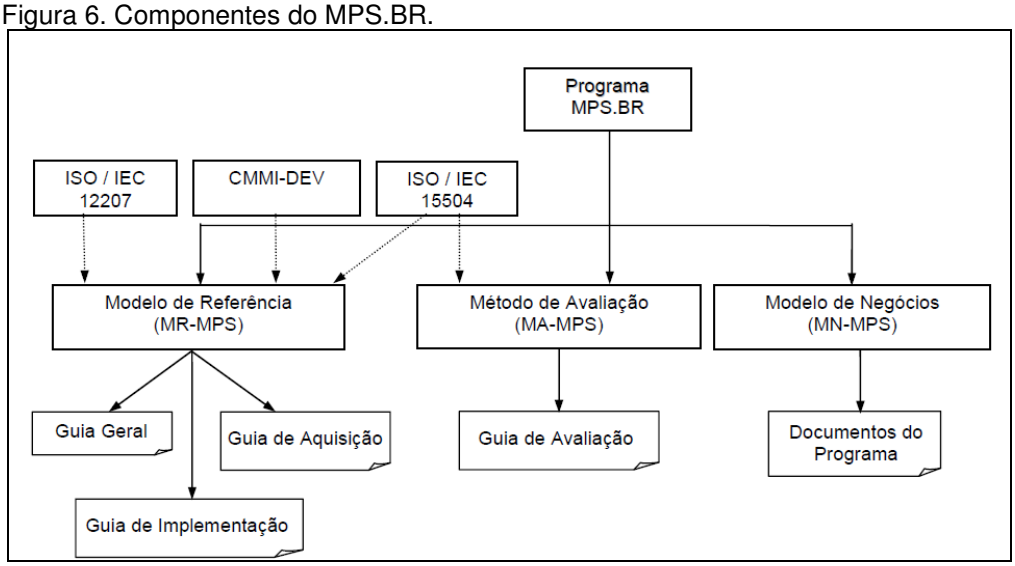
2.2 MODELO DE QUALIDADE MPS.BR

O MPS.BR baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos. Dentro desse contexto, o mesmo é dividido em 3 partes, como mostrados na figura 6: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS).

O MPS.BR está descrito por meio de documentos em formato de guias, incluindo:

a) **guia geral:** contém a descrição geral do MPS.BR e detalha o Modelo de Referência (MR-MPS), seus componentes e as definições comuns necessárias para seu entendimento e aplicação(MPS.BR, 2007a);
b) **guia de implementação:** composto de 7 partes, cada uma delas descrevendo como implementar um determinado nível do MR-MPS (MPS.BR, 2007d);
c) **guia de aquisição:** segundo MPS.BR (2007c), descreve um processo de aquisição de software e serviços correlatos. É descrito como forma de apoiar as instituições que queiram adquirir produtos de software e serviços correlatos apoiando-se no MR-MPS;
d) **guia de avaliação:** descreve o processo e o método de avaliação MA-MPS, os requisitos para avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA) (MPS.BR, 2007b).

Figura 6. Componentes do MPS.BR.



Fonte: SOFTEX (2009b)

O Modelo de Referência MR-MPS contém os requisitos que os processos das unidades organizacionais devem atender para estar em conformidade com o modelo. Ele contém as definições dos níveis de maturidade, processos e atributos do processo, e está descrito no Guia Geral (SOFTEX, 2009a).

O Guia de Aquisição que segue, é um documento complementar destinado a organizações que pretendam adquirir software e serviços correlatos. Este não contém requisitos do MR-MPS, mas boas práticas para a aquisição de software e serviços correlatos.

Segundo a Softex (2009a), o modelo de avaliação contém o processo e o método de avaliação MA-MPS, os requisitos para os avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA). O processo e o método de avaliação MA-MPS está em conformidade com a norma ISO/IEC 15504-2 (ISO/IEC 15504-2, 2003).

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

Nesse projeto será abordado detalhadamente somente o modelo de referência MR-MPS.

2.2.1 MR-MPS

O MR-MPS é definido por meio de níveis de maturidade, seqüenciais e acumulativos. Cada nível de maturidade é uma junção entre processos e capacidade dos processos, ou seja, é composto por um conjunto de processos em um determinado nível de capacidade. Os processos e as capacidades dos processos foram descritos segundo as normas ISO/IEC 12207 e ISO/IEC 15504-5. O progresso e o atendimento do nível de maturidade se obtêm quando são atendidos todos os resultados e propósito do processo, e os atributos de processo relacionados aquele nível.

A capacidade do processo é a caracterização da habilidade do processo para alcançar os objetivos de negócio, atuais e futuros; estando relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade.

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria de implementação de processos na organização.

De acordo com a Softex (2009c), o MR-MPS é formado por sete níveis progressivos de maturidade sendo que começa pelo nível G, o menos maduro, e encerra no nível A de maior maturidade, que são: Nível A **-** Em Otimização (mais maduro), Nível B - Gerenciado Quantitativamente, Nível C – Definido, Nível D **-** Largamente Definido, Nível E **-** Parcialmente Definido, Nível F **–** Gerenciado, Nível G **-** Parcialmente Gerenciado (inicial).

Os níveis de maturidade do MR-MPS seguem uma caracterização similar à dos quatro níveis de maturidade da representação por estágio do CMMI (níveis 2 a 5), sendo os níveis F, C, B e A do MR-MPS correspondentes respectivamente aos níveis 2, 3, 4 e 5 do CMMI. O nível G é um nível intermediário entre os níveis 1 e 2 do CMMI e os níveis E e D são dois níveis intermediários entre os níveis 2 e 3 do CMMI.

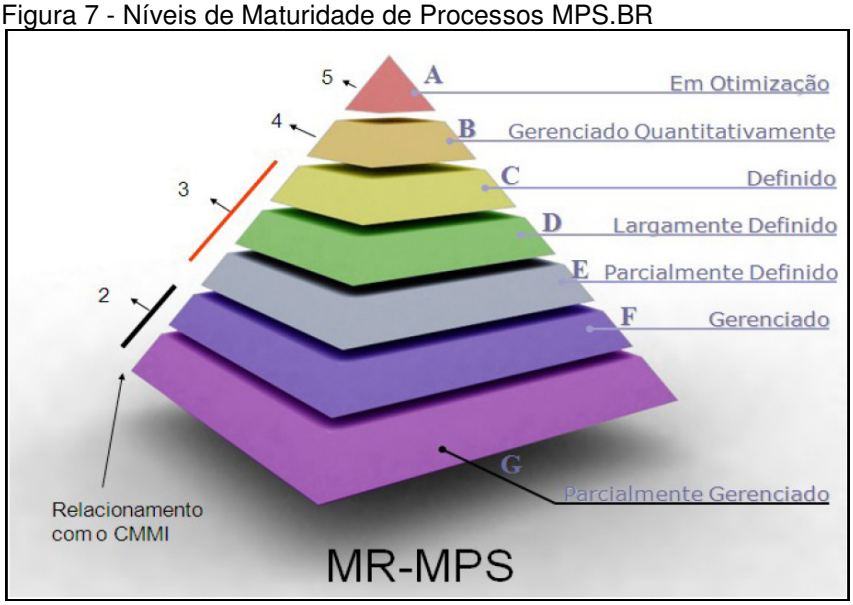
Esta divisão em sete níveis do MR-MPS possibilita uma implementação e reconhecimento gradual da melhoria de processo de software, facilitando sua adequação às pequenas e médias empresas, com visibilidade dos resultados em prazos mais curtos. A correspondência entre os níveis do MR-MPS e os do CMMI permite que um mesmo esforço de melhoria possa ser reconhecido pelo MR-MPS e pelo CMMI por meio de avaliações específicas.

2.2.1.1 Níveis de maturidade e seus processos

Nesse trabalho será aprofundado e serão apresentados os níveis de maturidades contendo também seus processos sendo que o nível D será detalhado.

A figura 7 mostra os sete níveis de maturidade que compõe o modelo MPS.BR.

Figura 7 - Níveis de Maturidade de Processos MPS.BR



Fonte: Softex (2009c)

O quadro 2 apresenta os níveis de maturidade do MR-MPS, os processos e os atributos de processo correspondente a cada nível.

Quadro 2 - Níveis de Maturidade do MR-MPS e seus Processos

Nível	Nome e Sigla dos Processos	Atributos de Processo (Capacidade)
A	Análise de Causas de Problemas e Resolução – ACP	AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2, AP 5.1 e AP 5.2
B	Gerência de Projeto – GPR (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1 e AP 4.2
C	Gerência de Riscos – GRI Desenvolvimento para Reutilização - DRU Análise de Decisão e Resolução – ADR Gerência de Reutilização – GRU (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
D	Verificação – VER Validação – VAL Projeto e Construção do Produto - PCP Integração do Produto – ITP Desenvolvimento de Requisitos – DRE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
E	Gerência de Projetos – GPR (evolução) Gerência de Reutilização – GRU Gerência de Recursos Humanos - GRH Definição do Processo Organizacional – DFP Avaliação e Melhoria do Processo Organizacional – AMP	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
F	Medição – MED Garantia de Qualidade – GQA Gerência de Configuração – GCO Aquisição – AQU	AP 1.1, AP 2.1 e AP 2.2
G	Gerência de Projeto – GPR Gerência de Requisitos – GRE	AP 1.1 e AP 2.1

Fonte: MPS.BR (2007a)

A seguir são especificados detalhadamente os níveis de maturidade com os processos e os atributos de processo correspondentes:

Nível G – Parcialmente Gerenciado

Os processos que este nível apresenta são:

a) **gerência de projetos – GPR:** tem como objetivo identificar, organizar, estabelecer e submeter o controle das atividades e recursos necessários para que o projeto apresente o resultado esperado, atendendo aos suas restrições e requisitos;
b) **gerência de requisitos – GRE:** seu propósito é gerenciar os requisitos do projeto, bem como a possíveis inconsistências entre os requisitos e os planos e também produtos de trabalho.

Nível F – Gerenciado

Os níveis desse processo são:

a) **gerência de configuração - GCO:** tem como objetivo manter a integridade dos produtos do projeto e gerenciar a disponibilidade a todos os envolvidos;
b) **garantia de qualidade – GQA:** seu propósito é garantir que os processos e produtos do trabalho e processos permaneçam nos moldes dos planos e requisitos definidos;
c) **medição – MED:** seu objetivo é recolher e especificar os dados relativos aos produtos produzidos e aos processos implementados na unidade organizacional e em seus projetos assim apoiando os objetivos organizacionais;
d) **aquisição – AQU:** tem como objetivo estabelecer critérios para aquisição de produtos e serviços que possam satisfazer o cliente.

Nível E – Parcialmente Definido

No nível E do processo os níveis de maturidade são:

a) **treinamento – TER:** sua finalidade é disponibilizar nos projetos da organização profissionais capacitados que executem as atividades de forma competente;
b) **definição do processo organizacional – DFP:** seu objetivo é estabelecer os processos ativos que serão empregados nas necessidades de negócio da organização;
c) **avaliação e melhoria do processo organizacional – AMP:** tem por finalidade instituir se os processos definidos na organização auxiliam na obtenção de resultados e no planejamento das ações futuras;
d) **adaptação do processo gerência de projetos – APG:** seu propósito é a gerência do projeto abrangendo os interessados através dos processos-padrão definidos.

Nível D – Largamente Definido

O nível D engloba os processos dos níveis G, F e E acrescidos dos processos de Desenvolvimento de Requisitos, Integração do Produto, Projeto e Construção do Produto, Validação e Verificação, cujos propósitos e resultados esperados serão descritos a seguir:

a) **desenvolvimento de requisitos – DRE:** a finalidade deste processo é estabelecer os requisitos dos componentes do produto e do cliente (MPS.BR,2007a):

- DRE 1: As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas,

- DRE 2: Um conjunto definido de requisitos do cliente é especificado a partir das necessidades, expectativas e restrições identificadas,

- DRE 3: Um conjunto de requisitos funcionais e não-funcionais, do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente,

- DRE 4: Os requisitos funcionais e não-funcionais de cada componente do produto são refinados, elaborados e alocados,

- DRE 5: Interfaces internas e externas do produto e de cada componente do produto são definidas,

- DRE 6: Conceitos operacionais e cenários são desenvolvidos,

- DRE 7: Os requisitos são analisados para assegurar que sejam necessários, corretos, testáveis e suficientes e para balancear as necessidades dos interessados com as restrições existentes,

- DRE 8: Os requisitos são validados;

b) **integração do produto – ITP:** sua finalidade é compor os componentes do produto produzindo um produto integrado consistente com o projeto demonstrando satisfação dos requisitos funcionais e não-funcionais para o ambiente equivalente (MPS.BR, 2007a):

- ITP 1: Uma estratégia de integração, consistente com o projeto e com os requisitos do produto, é desenvolvida para os componentes do produto,

- ITP 2: Um ambiente para integração dos componentes do produto é estabelecido e mantido,

- ITP 3: A compatibilidade das interfaces internas e externas dos componentes do produto é assegurada,

- ITP 4: As definições, o projeto e as mudanças nas interfaces internas e externas são gerenciados para o produto e os componentes do produto,

- ITP 5: Cada componente do produto é verificado, utilizando-se critérios definidos, para confirmar que estes estão prontos para a integração,

- ITP 6: Os componentes do produto são integrados, de acordo com a seqüência determinada e seguindo os procedimentos e critérios para integração,

- ITP 7: Os componentes do produto integrados são avaliados e os resultados da integração são registrados,

- ITP 8: Uma estratégia de regressão é desenvolvida e aplicada para uma nova verificação do produto, caso ocorra uma mudança nos componentes do produto (incluindo requisitos, projeto e códigos associados),

- ITP 9: O produto e a documentação relacionada são preparados e entregues ao cliente;

c) **validação – VAL:** MPS.BR (2007) afirma que o objetivo é confirmar que um produto atenderá a seu uso pretendido quando colocado no ambiente para o qual foi desenvolvido, e os resultados são:
- VAL 1: Produtos de trabalho a serem validados são identificados,
- VAL 2: Uma estratégia de validação é desenvolvida e implementada, estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação,
- VAL 3: Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido,
- VAL 4: Atividades de validação são executadas para garantir que os produtos de software estejam prontos para uso no ambiente operacional pretendido,
- VAL 5: Problemas são identificados e registrados,
- VAL 6: Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas,
- VAL 7: Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas;

d) **verificação – VER:** esse processo tem o propósito de confirmar que cada serviço e/ou produto de trabalho reflete apropriadamente os requisitos especificados. Seus resultados esperados são (MPS.BR, 2007a):
- VER 1: Produtos de trabalho a serem verificados são identificados,
- VER 2: Uma estratégia de verificação é desenvolvida e implementada, estabelecendo cronograma, revisores envolvidos, métodos para verificação e qualquer material a ser utilizado na verificação,
- VER 3: Critérios e procedimentos para verificação dos produtos de trabalho a serem verificados são identificados e um ambiente para verificação é estabelecido,

- VER 4: Atividades de verificação, incluindo testes e revisões por pares, são executadas,

- VER 5: Defeitos são identificados e registrados,

- VER 6: Resultados de atividades de verificação são analisados e disponibilizados para as partes interessadas;

e) **projeto e construção do produto – PCP**: tem como finalidade é projetar, desenvolver e implementar soluções para que os requisitos sejam atendidos. Segundo MPS.BR (2007a):

- PCP 1: Alternativas de solução e critérios de seleção são desenvolvidos para atender aos requisitos definidos,

- PCP 2: Soluções são selecionadas para o produto ou componentes do produto, com base em cenários definidos e em critérios identificados,

- PCP 3: O produto ou componente do produto é projetado e documentado,

- PCP 4: As interfaces entre os componentes do produto são projetadas com base em critérios predefinidos,

- PCP 5: Uma análise dos componentes do produto é conduzida para decidir sobre sua construção, compra ou reutilização,

- PCP 6: Os componentes do produto são implementados e verificados de acordo com o projeto (design),

- PCP 7: A documentação é identificada, desenvolvida e disponibilizada de acordo com os padrões identificados,

- PCP 8: A documentação é mantida de acordo com os critérios definidos;

Nível C – Definido

Os níveis que esse processo apresenta são:

a) **gerência de risco – GRI:** esse processo tem por objetivo gerenciar continuamente os riscos na organização;

b) **análise de decisão e resolução – ADR:** sua finalidade é definir processos formais para tomada de decisão de maneira a avaliar as alternativas identificadas.

Nível B – Gerenciado Quantitativamente

Esse nível de processo apresenta os níveis de maturidade a seguir:

a) **desempenho do processo organizacional – DEP:** o objetivo deste é apoiar as metas de desempenho do processo e qualidade através de uma análise quantitativa do desempenho do processo-padrão;
b) **gerência quantitativa do projeto – GQP:** seu propósito é gerenciar quantitativamente o projeto de forma a atender aos objetivos.

Nível A – Em Otimização

No nível A do processo os níveis de maturidade são:

a) **inovação e implantação da organização – IIO:** tem como objetivo selecionar e implantar inovações para a melhoria e aperfeiçoamento dos processos e tecnologias da organização;
b) **análise e resolução de causas – ARC:** sua finalidade é identificar a causa de problemas prevenindo ocorrências futuras.

Cada nível tem uma distribuição de atributos dos processos, esses que se acumulam em relação aos níveis, sendo que o nível F, por exemplo, tem os atributos do nível G, assim sucessivamente, portanto é obvio que para uma organização obter o nível F, é necessário obter primeiramente o G.

Os atributos dos processos são definidos como:

a) AP 1.1: O processo é executado;
b) AP 2.1: O processo é gerenciado;
c) AP 2.2: Os produtos de trabalho do processo são gerenciados;
d) AP 3.1: O processo é definido;
e) AP 3.2: O processo está implementado;
f) AP 4.1: O processo é medido;
g) AP 4.2: O processo é controlado;

h) AP 5.1: O processo é objeto de inovações;

i) AP 5.2: O processo é otimizado continuamente.

Sendo que para uma empresa que está no nível D, nível o qual é objetivo principal a ser abordado neste capítulo, contém os atributos de processo: AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2.

3 TESTES DE SOFTWARE

Para Rios (2007), testar é verificar se o software está atendendo às necessidades de acordo com seus requisitos e se não está fazendo o que não deveria fazer.

Teste de software, segundo Myers (2004), tem o objetivo de buscar falha durante a execução de um programa, e um bom caso de teste é aquele que tem alta probabilidade de conseguir desvendar erros que ainda não foram descobertos. Já um teste bem sucedido é aquele que revela o erro que ainda não foi encontrado.

O teste de software é a verificação do software com intuito de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve atuar, incluindo o processo de utilizar o produto para encontrar seus defeitos.

De modo generalizado teste de software garante que os riscos foram minimizados, as necessidades dos clientes foram acatadas, assegurar que os requisitos levantados na fase inicial foram realmente atendidos, e que os erros e defeitos ainda não descobertos sejam encontrados antes da fase de ratificação.

O desenvolvimento de software é acompanhado por uma atividade de garantia de qualidade (DEUTSCH apud PRESSMAN,1995).

> O desenvolvimento de sistemas de software envolve uma série de atividades de produção em que as oportunidades de injeção de falhas humanas são enormes. Erros podem acontecer logo no começo do processo, onde os objetivos podem estar errônea ou imperfeitamente especificados, além de erros que venham a ocorrer em fases de projeto e desenvolvimento posteriores. Por causa da incapacidade que os seres humanos têm de executar e comunicar com perfeição, o desenvolvimento de software é acompanhado por uma atividade de garantia de qualidade (DEUTSCH apud PRESSMAN, 1995, p. 786).

Para Delamaro, Maldonado e Jino (2007), a criação de um software não é nada fácil e necessita de muita minuciosidade, podendo ser cada vez mais complexo dependendo das características e dimensões do sistema a ser desenvolvido.

Devido a esse fato, inúmeros acontecimentos podem ocorrem ao final do software apresentando resultados diferentes do que o esperado.

Para garantir que os requisitos estejam em conformidade com a necessidade do cliente é de extrema importância que o software passe por um

processo de revisão, para que qualquer problema seja notificado antes que chegue ao cliente.

Pressman (2007), afirma que o processo de revisão, representado pela atividade de teste de software, é um elemento crítico para a garantia da qualidade final do sistema.

3.1 RISCO DE SOFTWARE

Rios (2007), define o risco como um evento no futuro cuja ocorrência poderá acarretar algum tipo de problema, no caso, o projeto, o produto ou o negócio.

Risco é a possibilidade de sofrer um impacto na qualidade do produto final, uma perda, um atraso do cronograma, um aumento nos custos ou falha do projeto (SEI, 2009).

Um risco nunca é uma certeza de ocorrência. Se alguma coisa vai realmente ocorrer com probabilidade de 100%, então isso não é um risco e sim um problema. Um risco pode ser tratado preventivamente, diferentemente de um problema.

Não há uma certeza de que vai ocorrer um determinado evento, porém, caso ocorra, uma perda poderá trazer sérios problemas. Para o teste de software, um problema seria deixar passar um erro no software por não ter testado um módulo importante.

Como exemplo pode-se citar que um navio sempre corre o risco de afundar, como o de um avião cair, mas só ocorrerá perda se realmente isso acontecer. É dever de a própria empresa checar e realizar todas as manutenções necessárias para que o risco de que um problema venha causar o naufrágio do navio seja sempre o menor.

No software por sua vez, ao realizar um processo de análise de risco, os esforços de teste vão ser priorizados nas áreas em que a ocorrência de um erro possa representar um grande impacto para o negócio ou para o cliente, dessa forma, como citado no exemplo acima, o objetivo é reduzir ao máximo a ocorrência de um erro que possa causar um grave impacto.

3.1.1 Análise de Risco

O teste de software está totalmente relacionado ao risco. Se por algum motivo um testador deixa de testar uma parte do sistema, ele está assumindo um risco já que parte do sistema não passará por teste. Se o risco por algum motivo vier a se materializar e a parte do sistema que não foi testada conter um erro, poderá acarretar à empresa uma grave perda financeira ou algum outro tipo de problema.

O risco é um dos elementos mais significativo a ser trabalhado no momento de elaborar um projeto de teste de uma aplicação (RIOS, 2007).

O gerenciamento de risco é um processo que tem por finalidade reduzir a ocorrência de prováveis eventos que poderiam ter um impacto negativo no final do produto.

Segundo Rios (2007) para se fazer uma análise de risco é necessária uma avaliação de alguns fatores como a probabilidade da ocorrência do risco e o impacto de haver uma perda ligada ao risco, sendo definido como a possibilidade de um erro em uma aplicação vezes o prejuízo caso esse erro se consolide como uma falha.

Ao planejar os testes, esses dois elementos devem ser levados em conta:

Risco	=	Probabilidade (possibilidade de falha)	X	Impacto (prejuízo causado pela falha)

Os riscos devem ser analisados principalmente pelos seguintes motivos:

a) para avaliar as consequências em caso de falha do produto;
b) para decidir as ações necessárias e certas antes e depois de o risco se materializar;
c) prevenir os impactos negativos;
d) proteger as vítimas (usuários e clientes);
e) para priorizar os riscos;
f) para ter critério e direção antes e durante a execução dos testes;
g) para fazer o melhor uso dos recursos investidos (materiais, humanos e tempo);
h) conhecer as áreas que precisam de maior atenção;

i) conhecer com antecedência o que pode dar errado.

De um modo geral, ao analisar os riscos no software com os devidos procedimentos, é possível obter um fundamento importante sobre as áreas e recursos que serão suficientes para conduzir um bom teste no software, evitando perdas de recursos e dando mais ênfase na parte do sistema mais crítico.

3.1.2 Classificação de Riscos

Os principais riscos que possam existir em um projeto de teste de software segundo Rios (2007) são:

a) riscos de projeto, estes que são diretamente ligados ao projeto, como problemas com cronograma, recursos, pessoal, requisitos, cliente, orçamento;
b) o riscos técnicos, são risco ligados à qualidade do software a ser desenvolvido, como problemas com a implementação, interface, defeitos no software, ambigüidade de especificação;
c) riscos para o negócio do cliente, são aqueles que afetam as organizações envolvidas no negócio, sejam por prazo de desenvolvimento, mudanças no orçamento ou problemas de mercado;
d) riscos de processo são associados ao processo de negócio ou outro processo que possa impactar a organização, o usuário (cliente) ou o projeto.

3.2 VERIFICAÇÃO

A atividade de teste é dividida em duas áreas chamadas de validação e verificação (V&V), essas áreas são responsáveis por assegurar que o software atenda às necessidades dos usuários e cumpram com suas especificações.

Seus objetivos principais são descobrir defeitos no sistema e assegurar se o sistema é ou não utilizável em uma situação operacional. Para Koscianski e Soares (2007), essas áreas constituem em uma série de tarefas, que tem início com as revisões dos requisitos, passando pelas revisões das análises e do projeto de software e as inspeções do código até chegar aos testes.

A verificação se define como um conjunto de atividades que determinam se os requisitos especificados foram efetivamente atendidos no sistema, ou seja, avaliar se o software realmente cumpre as especificações necessárias.

A atividade de verificação tem por objetivo analisar se o que foi planejado realmente foi alcançado, se os requisitos, funcionalidades e desempenho documentados foram implementados e ainda prever falhas ou inconsistências entre requisitos.

Pressman (2007) coloca em questão uma frase para definir a verificação: “Estamos construindo certo o produto?”. A diferença se mostra na questão que define validação: “Estamos construindo o produto certo?”, que será conceituada a seguir.

3.3 VALIDAÇÃO

A validação é um conjunto de atividades que garantem que a especificação de uma fase ou sistema completo consista com os requisitos do cliente e se é apropriada, pois o software deve estar de acordo com as exigências e desejos dos usuários. Suas principais atividades a serem contempladas são: a revisão dos requisitos, inspeções do código, revisões do projeto e testes do sistema.

Rios (2007), cita exemplos das atividades de V & V:

a) **atividades de verificação:** revisões de requisitos, revisões de modelos, revisões de código e inspeções técnicas em geral;
b) **atividades de validação:** teste integração, teste de sistemas, teste unitário, teste de aceitação e teste de homologação.

As atividades de (V&V) estão distribuídas por todas as etapas do processo de teste, cada atividade apresenta duas características diferentes: testes estáticos e testes dinâmicos.

A atividade de validação é executada após da atividade de verificação, os documentos são verificados antes da execução dos testes, por essa razão suas atividades têm a característica de testes estáticos já que nessa atividade o software não é executado. Já a atividade de validação, as técnicas utilizadas nessa fase utilizam à execução do software e são caracterizadas como testes dinâmicos.

4 PROCESSOS DE TESTE DE SOFTWARE

Não é possível idealizar um processo de teste de software sem integrá-lo com o ciclo de desenvolvimento de um software. Um bom processo de teste é aquele que cria uma relação de "um-para-um" entre as fases de desenvolvimento e testes. Esta relação bilateral promove a colaboração entre as áreas e reforça a idéia do objetivo comum.

Os resultados mais expressivos das atividades de testes só começam a aparecer depois que a atividade de teste começa a ser tratada como um processo.

O Processo de testes representa uma estrutura das etapas, artefatos, atividades, papéis e responsabilidades buscando padronizar os trabalhos para ter melhor controle dos projetos de testes. O objetivo deste é minimizar os riscos causados por defeitos derivados do processo de desenvolvimento e também a redução de custos de correção de defeitos, pois os custos do software com desenvolvimento e a manutenção tendem a ser menor quando o software é bem testado.

4.1 CONCEITOS

O teste do software é um processo realizado pelo testador de softwares que atravessa outros processos da engenharia de software, envolvendo ações que vão do levantamento de requisitos até a execução do teste propriamente dito. O objetivo é encontrar defeitos nos produtos, para que estes possam ser corrigidos antes da entrega final do produto. A maioria das pessoas pensa que o teste de software tem como finalidade demonstrar o correto funcionamento de um programa, sendo que na verdade é utilizado como um processo da engenharia de software designado à encontrar defeitos.

O processo de teste de software é voltado para o alcance de um nível de qualidade de determinado produto, que durante o processo de desenvolvimento muda conforme avanço das atividades, requisitos, modelo de dados lógico, modelo de dados físico, protótipos, código-fonte, módulos funcionais e finalmente um sistema.

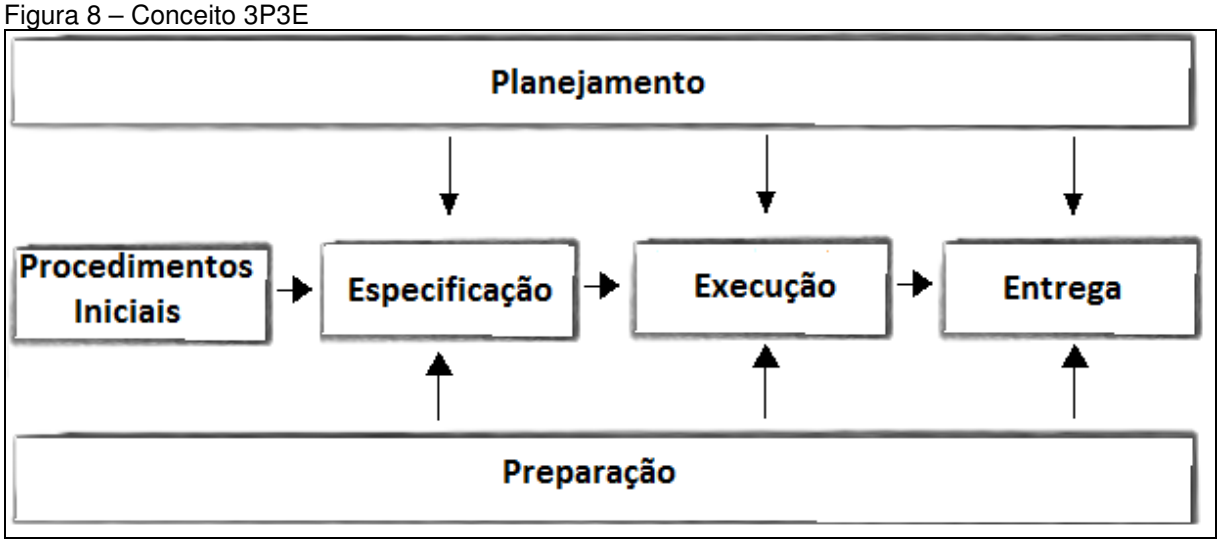
Existem várias definições para esse conceito de teste. De forma simples, o conceito de teste de software pode ser definido de tal modo: testar um software

significa verificar através de uma execução controlada se o comportamento está de acordo com o especificado. O objetivo principal desta atividade é encontrar o número máximo de erros utilizando-se do mínimo de esforço, ou seja, mostrar aos desenvolvedores se os resultados estão ou não de acordo com os padrões estabelecidos.

4.2 CICLO DE VIDA DO PROCESSO DE TESTE DE SOFTWARE

O ciclo de vida de testes é composto pelas seguintes etapas: planejamento, preparação, procedimentos iniciais, especificação, execução, entrega conforme mostrado na figura 8 sendo quatro dessas etapas seqüenciais ou em cascata e duas paralelas.

Figura 8 – Conceito 3P3E



Fonte: Rios e Moreira Filho (2006. P. 9)

Planejamento é a etapa onde se estabelece o que vai ser testado, em quanto tempo e em que momento os testes serão interrompidos.

Procedimentos iniciais onde os objetivos do processo de software são definidos; requisitos do negócio, atividades a serem executadas, recursos humanos e recursos de ambiente.

Na fase de especificação a atividade principal é elaborar e revisar os cenários e roteiros de testes, criação dos casos de teste.

Já na fase de execução executam-se os testes planejados, casos de teste, e se registra os resultados obtidos.

Entrega é onde se arquiva toda a documentação e se descrevem todas as ocorrências do projeto relevantes para a melhoria do processo, finalização do processo de teste com entrega do sistema.

Preparação onde o objetivo é preparar toda a estrutura do ambiente de testes como: configuração de hardware e softwares usados (sistemas operacionais, browsers, etc.), equipamentos, criação da massa de dados de teste, pessoal, ferramentas de automação.

4.2.1 Modelos de Teste

Um modelo de teste é um conjunto de técnicas que representam as dimensões do teste, ou seja, em qual fase do desenvolvimento de um projeto se aplica um determinado teste.

De acordo com Rios (2007), as principais técnicas de teste utilizadas são:

a) **teste de unidade:** essa técnica costuma ser executada pelos próprios desenvolvedores com objetivo de testar os componentes do software visando garantir que as unidades de código, para que as especificações do sistema sejam atendidas;

b) **teste de integração:** visa garantir que os componentes do sistema funcionem de maneira correta;

c) **teste de sistema:** execução do sistema como um todo, tem seu teste voltado para validar a perfeição e a exatidão na execução das funções do sistema.

d) **teste de aceitação:** é a última ação de teste antes da implementação do software, sua execução é de responsabilidade do cliente. A finalidade desse teste é verificar se o sistema está atendendo as especificações que o cliente solicitou.

O modelo de teste em “V” é muito prático e seu conceito simples, é modelo mais popular entre todos existentes (MOLINARI, 2008). Do lado esquerdo do “V” se encontra o ciclo de desenvolvimento do software e ao lado direito a parte do processo de teste, que para Bastos et al (2007), são:

a) **acesso ao plano de desenvolvimento:** pré-requisito para a elaboração do plano de testes. Nessa fase é feita uma verificação do

plano de desenvolvimento e é possível avaliar os recursos necessários para a realização dos testes;

b) **desenvolvimento do plano de testes:** segue os mesmos modelos do plano de desenvolvimento, sua estrutura varia conforme os riscos associados com o software que está sendo desenvolvido;
c) **inspeção ou teste dos requisitos do software:** através da técnica de verificação, os requisitos do software são avaliados. Essa fase é muito importante, pois requisitos mal analisados representam a maioria dos insucessos do desenvolvimento do software;
d) **inspeção ou teste do desenho do software:** por meio da técnica de verificação, o desenho do software é verificado de modo que garanta que o objetivo dos requisitos foi alcançado no desenho do software;
e) **inspeção ou teste da construção do software:** a partir do desenho do sistema é determinado a extensão e o tipo dos testes que serão necessários. Quanto mais a construção se torna automatizada, menos testes serão solicitados durante esta fase;
f) **execução dos testes:** o código do sistema é testado dinamicamente, as ferramentas e abordagens especificadas no plano de teste serão usadas para validar se os requisitos do desenho do sistema foram realmente implementados;
g) **teste de aceitação**: a fase de validação do software feita pelos usuários. Além dos requisitos documentados, os usuários geralmente testam outras funções não documentadas. É necessário nessa fase analisar os possíveis erros encontrados pelo usuário já que esses erros podem ser tratados como mudanças e não como erro;
h) **informação dos resultados dos testes:** os resultados do teste devem ser armazenados em um relatório ou documento de teste que contenha os defeitos encontrados no processo. Esse documento deve ser informado aos setores envolvidos o mais rápido possível, de forma que as correções sejam feitas com custo menor possível;
i) **teste da instalação do software:** tendem à verificar a compatibilidade do sistema operacional com os procedimentos operacionais do software. O resultado vai determinar se o software está em condições de ser ou não implantado no ambiente de produção;

j) **teste das mudanças no software:** essa fase cobre as mudanças durante o processo de implementação e aquelas que irão ocorrer após o software estar implantado;
k) **avaliação da eficácia dos testes:** após o término dos testes, deve ser feito uma avaliação para verificar a eficácia dos mesmos. Esse processo tem que ser realizado pelos testadores, porém os desenvolvedores, usuários e outros profissionais podem ser envolvidos.

5 DEFINIÇÃO DE PROCESSO DE TESTE DE SOFTWARE

Para Molinari (2008), um processo de engenharia de software se define como um conjunto de passos parcialmente ordenados, cuja finalidade é atingir uma meta para que o produto final de software seja entregue com eficiência, previsível e que vá de encontro às necessidades de negócio.

De acordo com MPS.BR (2007), um processo de teste tem objetivo de alcançar as seguintes definições:

a) diminuição do retrabalho;

b) aumento da qualidade do produto;

c) maior produtividade;

d) redução do tempo para atender o mercado;

e) maior precisão nas estimativas;

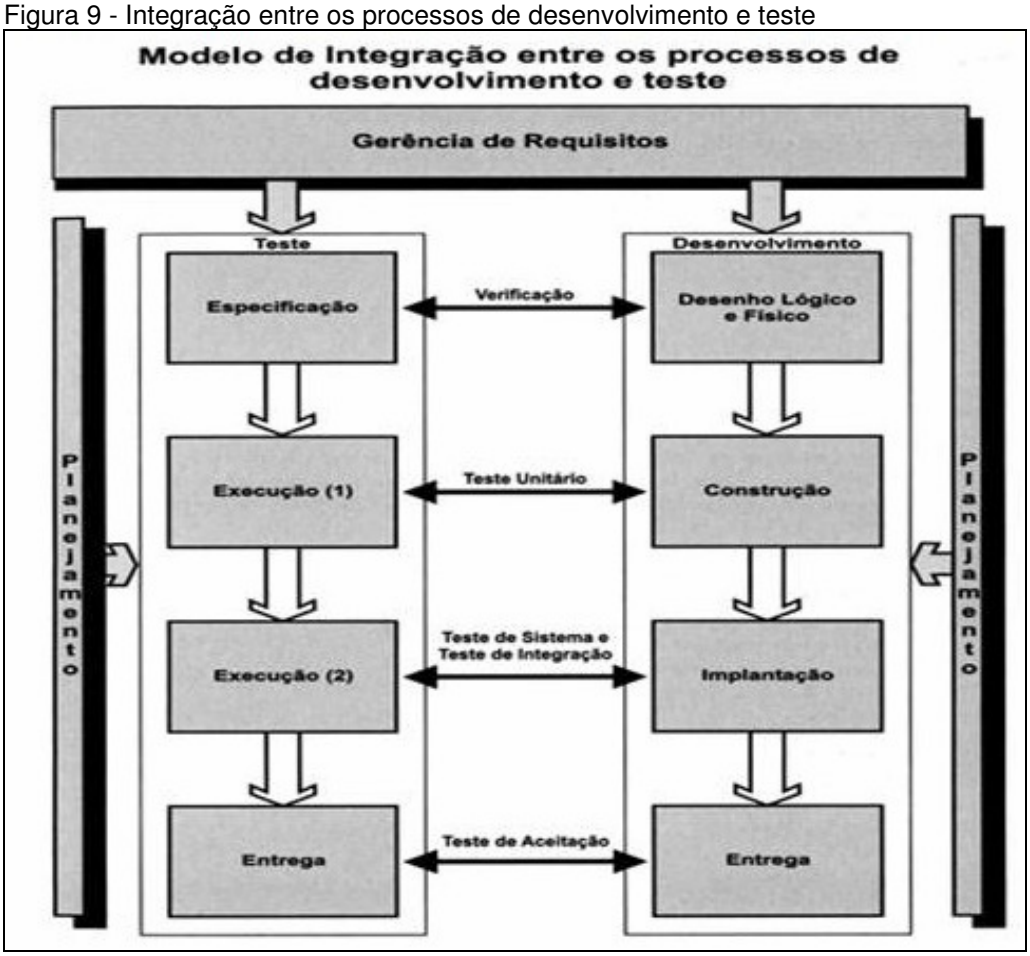
f) maior competitividade;

g) acompanhamento da satisfação do cliente.

Os testes passaram a ser tratados como processo próprio não mais como uma atividade dentro do processo de desenvolvimento. Os testes passaram a ter metodologia própria e serem executados junto com o início do processo de desenvolvimento, quanto antes serem executados os testes mais barato fica a correção dos possíveis defeitos encontrados.

A figura 9 apresenta o modelo de integração entre os processos de desenvolvimento e teste segundo Rios (2007).

Figura 9 - Integração entre os processos de desenvolvimento e teste



Fonte: Rios (2007, P.102)

Segue as fases e suas particularidades:

a) **desenho lógico é físico:** os testes de verificação asseguram nesta fase que o desenho do sistema esteja de acordo com os requisitos especificados;
b) **construção:** os testes unitários são executados em paralelo a codificação do sistema e os componentes desenvolvidos podem ser testados através dos testes unitários;
c) **implantação:** nessa fase o sistema já está codificado e suas funcionalidades lógicas já em funcionamento. São executados os testes de sistema e de integração;
d) **entrega:** os testes de aceitação são planejados e executados juntamente com o cliente.

6 TESTE DE SOFTWARE BASEADO EM RISCO

Ao dar início a um projeto é preciso ficar atento a possíveis acontecimentos futuros. Problemas com prazos de entrega, as exigências dos usuários não serem atendidas, problemas técnicos com equipamentos, entre outros, estes são problemas que podem ocorrer, sendo que o produto final de qualquer forma tem que ser entregue sem falhas e com qualidade. Portanto, é importante se prevenir destes tipos de eventos inesperados.

Esse capítulo tem como objetivo relacionar riscos com testes de software, o que é chamado de *Risk-based Testing,* testes baseados em riscos.

James Bach publicou em 1995 um artigo chamado: The Challenge of Good Enough na revista American Programmer. Nesse artigo, Bach apresentou uma nova abordagem de teste, que foi declarada teste baseado em risco, risk-based testing. Essa abordagem consiste em um conjunto de atividades que proporciona a identificação de fatores de riscos associados aos requisitos do produto de software.

Para materializar esse conceito, segundo Molinari (2008), é importante levar em conta três situações:

a) é inviável e impossível testar todas as funções de um sistema. Nessa situação é importante que o teste seja bem planejado, ou seja, criar um plano de teste que contenha a cobertura que será explorada, e as condições de início e final do processo de teste;

b) os prazos para o desenvolvimento de software na maioria das vezes não são cumpridos, sendo que nesse caso é necessário que sejam priorizadas as partes mais importantes do sistema em questão;

c) para que se obtenha sucesso na execução dessa abordagem é importante que o testador tenha um bom conhecimento da aplicação, sendo que para o software ser bem feito o profissional tem que ter plenos conhecimentos sobre o sistema a ser testado.

A partir do momento em que os riscos são priorizados, os casos de testes são gerados, baseados nas estratégias de acompanhamento e tratamento dos fatores de riscos que foram previamente identificados.

A abordagem dos testes baseados em riscos de acordo com Bach (1999), foca fazer julgamento sobre cobertura de teste, o número de testes a ser conduzido, as escolhas dos tipos de testes e de revisões, uso e balanceamento entre testes,

inspeções e revisões, dentre outros fatores e priorização dos testes (planejamento e execução).

Testar é elemento fundamental para desenvolver software de alta qualidade e garantir a tranqüilidade das operações, pois as conseqüências que algumas falhas podem ocasionar são terríveis. Além do mais, testes também trazem benefícios como a economia de tempo e de custos.

6.1 DEFINIÇÕES DE TESTE DE SOFTWARE BASEADO EM RISCO

O Risk-based Testing pode ser considerado uma técnica de teste que tem como finalidade analisar o software e realizar o planejamento dos testes a partir das áreas onde há probabilidade de ocorrer uma falha ou problema, essas que gerem maiores conseqüências.

Segundo Bach (1999), é utilizado a definição das etapas e atividades do processo de teste baseado em risco como segue abaixo:

a) **listar os riscos de acordo com suas prioridades:** identificar e analisar os riscos do software;
b) **elaborar testes que explorem cada risco:** criar e executar testes que verifiquem a existência dos riscos identificados;
c) **gerenciar os riscos:** assim que eliminar um risco, um novo risco pode surgir sendo que os esforços de testes devem ser reajustados para que os novos riscos sejam tratados de acordo com suas prioridades.

Apesar de simples, essa definição consegue discutir a idéia principal da abordagem de teste de software baseada em risco.

Quando um projeto é avaliado por um analista, o mesmo busca aqueles fatos cujas ocorrências poderão acarretar em danos para a empresa. Nesta abordagem um produto é avaliado utilizando-se as seguintes perdas:

a) **vulnerabilidade:** quais as possíveis falhas existentes neste elemento?
b) **ameaça:** que entradas ou situações poderiam existir que podem explorar uma vulnerabilidade e disparar uma falha neste componente?
c) **vítimas:** quem ou o quê poderia ser impactado por uma possível falha e quão ruim seria isto?

Tendo essas informações levantadas, o analista de teste tem os auxílios necessários para criar casos de testes específicos, que irá abordar o sistema em

sua parte mais vulnerável. Sendo assim é possível definir se será necessário escrever um caso de teste para uma determinada situação.

De forma geral o processo de teste de software baseado em riscos se descreve pelo processo de identificar os riscos de um sistema, analisá-los, priorizá-los, criar casos de teste que mitiga esses riscos identificados e gerencia sua execução.

7 TRABALHOS RELACIONADOS

Esse capítulo tem com finalidade apresentar projetos com características e objetivos semelhantes ao objetivo desta pesquisa, porém com seu foco desviado à outros interesses.

7.1 A IMPORTÂNCIA DO PROCESSO DE TESTE PARA A QUALIDADE DO SOFTWARE

Esta monografia foi proposta pela acadêmica Tilene Corsi Cavalcanti no curso de Tecnologia em Informática para Gestão de Negócios, da Faculdade de Tecnologia da Zona Leste, no ano de 2009, para obtenção do título de Tecnólogo em Informática para Gestão de Negócios, tendo como professor Msc. Antônio Tadeu Pellison.

O objetivo deste estudo foi evidenciar que a tecnologia mais do que nunca está embutida no mundo corporativo e na vida das pessoas, e que para auxilar as empresas nos processos evolutivos diversos métodos estão disponíveis, sendo a empresa responsável por analisar qual método realmente está de acordo. Foi proposto então que a atividade de teste de software seja um dos principais métodos para garantir a qualidade, foi realizado estudo de casos em uma empresa e feito comparativos para mostrar como é importante a realização de testes de software para a qualidade do produto final.

7.2 TESTE DE SOFTWARE BASEADO EM RISCO

Esta monografia foi proposta pelo acadêmico do curso de Ciência da Computação, Edson Andrade de Moraes, pela Pontifícia Universidade Católica do Rio de Janeiro, em março de 2006.

Seu objetivo principal foi abordar diferentes metodologias propostas, enfocando os ganhos que são defendidos por seus autores, a aplicabilidade e restrições de uso. É uma pesquisa bibliográfica, feita a partir de artigos.

7.3 MODELAGEM DE UM PROCESSO PARA O GERENCIAMENTO E DESENVOLVIMENTO DE REQUISITOS BASEADO NO MODELO DE QUALIDADE DE PROCESSO MPS.BR

Esse Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, pela acadêmica Ariela Dal Toe da Silva, tendo como orientador o professor Msc. Gustavo Bisognin, no ano de 2010.

O objetivo do referente trabalho é abordar o nível G do modelo de qualidade MPS.BR, e a criação de uma modelagem de processo para o gerenciamento e desenvolvimento de requisitos de software, com intuito de efetuar uma reciclagem das solicitações feitas pelos clientes durante todo o processo de desenvolvimento até a implantação de um software, minimizando assim o desperdício de tempo e dinheiro. Para modelagem do processo foi utilizada a ferramenta EPF Composer.

8. APLICAÇÃO DAS NORMAS DE QUALIDADE MPS.BR EM UMA MODELAGEM DE PROCESSO DE TESTE BASEADO EM RISCO

O mercado de software está muito concorrido e com isso as empresas estão em constante busca de melhorias, principalmente no que diz respeito a modelo de qualidade de processo.

O desenvolvimento de produtos de software está em constante evolução, e as empresas desenvolvedoras vêm enfrentando problemas quanto ao usuário final do software, e o nível de exigência aumenta a cada dia por parte dos clientes.

Para que um produto seja entregue ao usuário final, ele deve estar em perfeitas condições, sem falhas ou qualquer tipo de evento inesperado.

Apesar dos testes serem atividades difíceis e de alto custo para ser realizado em uma empresa devido aos inúmeros erros possíveis, os mesmos se tornaram parte essencial no desenvolvimento de um software. Se a área de teste não estiver bem organizada os defeitos vão persistir a ocorrer num estágio onde os custos são cada vez maiores.

Com intuito de minimizar problemas como estes esse projeto busca demonstrar um modelo de processo que facilita a execução da etapa de testes de software de uma empresa, sendo que os mesmos passam a ser executados por equipes especializadas.

O presente trabalho tem a finalidade de modelar processos de teste de software aplicado juntamente a normas de qualidade MPS.BR, mais especificamente no nível D, este que engloba a parte de validação a qual será avaliada no processo para saber se o mesmo está ou não apropriado as normas exigidas.

O desenvolvimento do projeto consiste no estudo de modelos de qualidade de software, no estudo da ferramenta EPF Composer, no desenvolvimento dos processos de teste baseados em risco e por fim a análise e os resultados que foram obtidos do ínicio ao fim do projeto.

A ferramenta base para criação de processos de teste baseados em risco foi o Eclipse Process Framework, EPF Composer, que é uma ferramenta de código aberto e gratuita que proporciona uma estrutura para gestão de processo de software através de um desenvolvimento ágil, iterativo e incremental, otimizado pra pequenos projetos. O mesmo dispõe produzir softwares com características

próprias, fornecendo conteúdo e ferramentas que podem ser utilizados como base para uma grande variedade de processos de TI.

Essa ferramenta auxilia na autoria de métodos, modelagem de processos, ferramentas, gestão de configuração de bibliotecas de métodos e publicação de processos.

Primeiramente foi realizado um levantamento bibliográfico a fim de se obter referencial teórico, em seguida voltou-se o estudo a normas e qualidade de software.

A segunda etapa foi criação dos processos de teste baseados em risco, para isso foram realizadas diversas pesquisas sobre testes de software, para poder entender e aplicar seus conceito, e também posteriormente poder aplicar normas de qualidade.

A abordagem de teste de software baseada em risco em geral consiste em um conjunto de atividades que facilitam a identificação de fatores de riscos associados aos requisitos do software.

Existem várias teorias relacionadas a teste de software baseados em risco, como a teoria de Bach que é considerado o pai dessa abordagem, ele define duas abordagens baseadas em heurística para identificar riscos. A primeira, “Inside-Out”, o que pode dar errado no produto é questionado e estudado repetidas vezes. A segunda, “Outside-In”, consiste na utilização de três listas de potenciais riscos juntamente com descrições do produto. Para cada risco, é identificado se o mesmo se aplica ou não a um determinado componente. Valores são atribuídos através de uma escala de interesse, os riscos com valores maiores são testados primeiro. Nessa abordagem há dificuldades, pois haverá de ter um conhecimento prévio do negócio a fim de realizar a análise dos riscos da maneira mais fiel possível.

Já na teoria de Besson a abordagem é baseada em uso, onde qualquer atividade de teste implica numa diminuição de risco, a idéia é investir o mínimo de esforço possível em teste para maximizar a redução dos riscos. O controle do esforço de teste baseado na utilização do software seguindo a teoria de Pareto, que afirma que 20% das funcionalidades permitem ao usuário realizar 80% do seu trabalho. Como nessa hipótese a execução de teste é baseada em esforço os teste que gastam menos tempo são executados primeiro, sendo que essa abordagem pode ser útil em culturas avessas ao teste.

Outros autores fornecem uma abordagem que através da aplicação de métricas de complexidade de software orientados a objetos, podem-se chegar às classes que possuem maior probabilidade de falhas, contudo os autores não propõem estratégias de teste para código muito menos formas de rastreamento das funcionalidades impactadas por classes com alto risco de falhas.

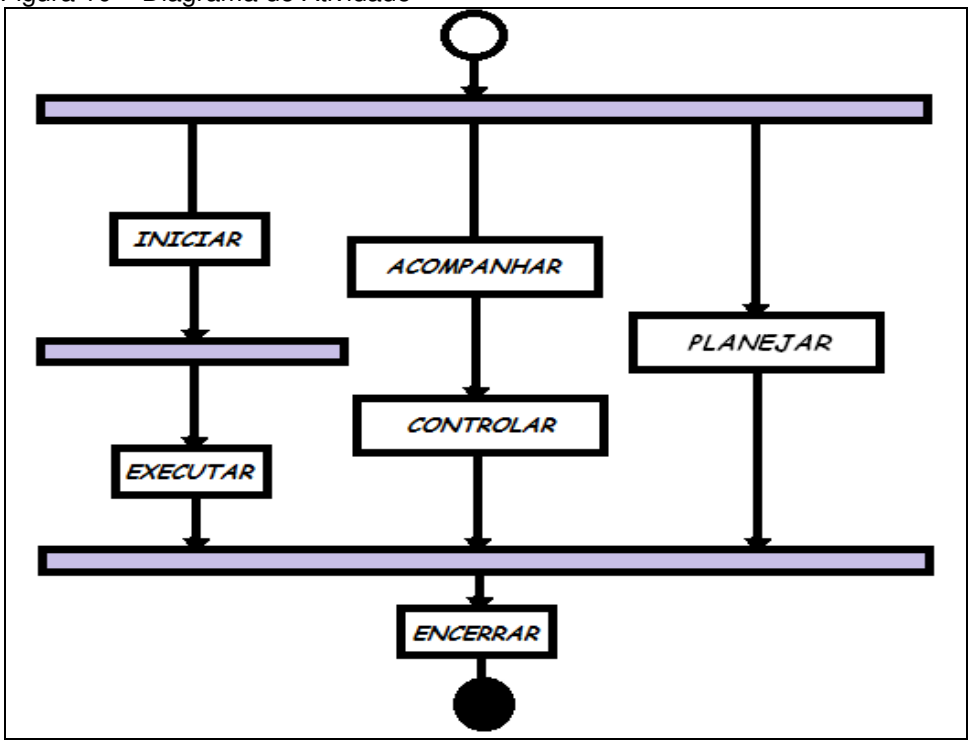
Dentre todas essas abordagens a mais completa que fornece guias, métricas, artefatos, papéis e atividades é o RBTProcess, processos de teste de software baseados em risco.

8.1 MODELAGEM DO PROCESSO DE TESTE DE SOFTWARE BASEADO EM RISCO

Esse processo é formado a partir da identificação das principais atividades presentes no gerenciamento de riscos e no processo de teste de software padrão, as atividades de software também sofreram mudanças com relação ao padrão, pois agora passam a ser guiadas por risco.

As fases dessa abordagem podem ser observadas na figura 10.

Figura 10 – Diagrama de Atividade



Os processos de teste baseados em risco, como mostra no diagrama possuem fases distintas, fase de iniciação, acompanhamento, controle, planejamento, execução o enceramento do mesmo.

Cada uma dessas fases desempenha uma atividade que é controlada e de extrema responsabilidade de um ator, ou seja, é realizada por um membro da equipe de testes.

Na primeira, fase iniciar, é necessário que sejam identificadas as necessidades e levantadas as informações e estimativas para a elaboração de um plano que contenha os principais objetivos e atividades que serão executados.

A fase de planejamento tem como foco a elaboração inicial dos testes com base na análise dos riscos. É nessa fase que serão elaboradas estratégias e o plano de teste, como também será descrito um caminho a ser seguido nas próximas etapas do processo, além de estabelecer o que vai ser testado, quando serão executados e em quanto tempo. Essa fase deve ser mantida ativa até que o projeto seja concluído, com a finalidade de avaliar se a mesma foi desenvolvida conforme o planejado. O marco desta fase é a priorização dos requisitos.

A fase seguinte, de acompanhamento, tem como objetivo acompanhar todo processo para detecção de possíveis defeitos e falhas, acompanha o processo desde o início até o fim.

Os resultados da execução dos testes são coletados e controlados na fase de controle. Compreende a atividade de avaliar testes onde são verificados números, estratégias, tempo de execução, solicitação de mudanças dos testes.

Na fase executar, os casos de testes planejados e projetados são executados. Abrange a atividade executar testes. O marco desta fase é a execução de casos de testes que verificam a existência ou não dos riscos identificados.

Por último na fase de encerramento compreende a atividade controlar risco, onde o acompanhamento dos riscos é realizado e tem como objetivo obter a aceitação final, avaliar e relatar as lições aprendidas na execução do projeto, gerar estimativa e garantir a qualidade do produto. Seu cumprimento é necessário para que o software possa ser liberado com uma margem segura e comprovada através de indicadores que a qualidade e a ausência de defeitos críticos estejam totalmente corrigidas. Segundo Souza, Gusmão e Rocha (2008) os riscos mitigados são eliminados da lista de riscos identificados e serve de entrada para o planejamento da próxima iteração. O marco desta fase é o controle dos riscos mitigados.

Cada fase do processo necessita de um membro, uma pessoa da equipe de teste, que tem um papel fundamental na execução de uma atividade.

8.1.1 Papéis executados na abordagem de teste de software baseados em risco

Fazem parte da equipe de teste o analista de risco, gerente de risco, projetista de teste e testador, com exceção do analista de risco que é um papel da gerencia de risco e essencial para identificação, análise e controle dos riscos, os outros papéis executados são os mesmos de um processo de teste padrão.

O analista é um membro da equipe de testes, um engenheiro de testes, este possui conhecimento sobre teste de software, identificação e análise dos riscos, além de profundo conhecimento sobre os requisitos do software.

Já para o gerente de testes são necessários alguns conhecimentos no que se refere à processos de testes, análise e planejamento e acompanhamento.

O projetista de testes necessita de um conhecimento aprofundado sobre os requisitos do sistema e tecnologia adotada, sobre os riscos, ferramentas, técnicas para construção, estratégias e tipos de testes.

O testador carece de conhecimento sobre o processo de teste, ferramentas utilizadas para execução, configuração do ambiente de execução e solicitação de mudança, além do conhecimento sobre os requisitos do sistema.

A seguir será especificada cada atividade desempenhada.

8.1.2 Atividades realizadas na abordagem de teste de software baseados em risco

As atividades têm como objetivo definir como os testes serão conduzidos no projeto, são elas: identificar riscos, analisar riscos, controlar riscos, essas que são provenientes do gerenciamento de riscos, planejar testes, projetar testes, executar testes e avaliar testes.

8.1.2.1 Identificar Riscos

Essa atividade é proveniente do gerenciamento de risco e tem como objetivo identificar os riscos técnicos associados aos requisitos do sistema que podem de alguma forma afetar o sucesso do projeto e que também podem ser mitigados através da execução de testes e documentar suas características.

De acordo PMI (2004) no PMBOK essa atividade pode ser dividida em três categorias que auxilia na identificação dos riscos que será exposta a seguir:

a) **revisar fontes e categoria de riscos**: nessa etapa o objetivo é avaliar e adequar as listas de riscos e o questionário para o software que será testado;
b) **responder questionário:** é a primeira etapa para a identificação dos riscos. Para casa requisito, são feitas algumas perguntas e respondidas por participantes e que de acordo com a resposta, é indicada a existência do risco ou não;
c) **realizar reunião de Brainstorm:** tem como objetivo validar os riscos levantados na atividade anterior. Para guiar a reunião são utilizadas as listas de risco. As reuniões de brainstorm com listas de riscos, visam tornar a reunião mais eficiente. Como produto dessa atividade tem-se uma lista de riscos identificados para cada requisito.

8.1.2.2 Analisar Riscos

Tem como finalidade a priorização dos requisitos a serem testados com base na análise dos riscos técnicos identificados, mede a probabilidade e as conseqüências dos riscos e estima suas implicações para os objetivos do projeto. É nessa etapa que é calculado a exposição do risco e analisado.

8.1.2.2.1 Calcular a exposição do risco

Para se fazer uma análise de risco é necessária uma avaliação de alguns fatores, como a probabilidade da ocorrência do risco e o impacto de haver uma perda ligada ao risco. A fórmula abaixo foi proposta por Rios (2007).

R = Pb x I **(1)**

Sendo R a exposição ao risco, Pb a probabilidade de ocorrência do risco vezes I impacto do risco.

Para que se possa evitar, mitigar ou aceitar um risco é necessário que o mesmo seja calculado utilizando a fórmula, para se saber o qual e quanto esse risco deve ser analisado.

Os valores que serão utilizados para calcular são propostos pela própria autora do trabalho e serão expostos em uma matriz de risco.

O impacto pode ser classificado de acordo com a gravidade:

a) **baixa:** impacto é irrelevante para o projeto, podendo ser resolvido facilmente;

b) **média:** impacto é relevante para o projeto e necessita de um gerenciamento mais preciso. Pode prejudicar o resultado do projeto;

c) **alta:** impacto extremamente elevado e os resultados poderão ser comprometidos.

Por meio da matriz de probabilidade de ocorrência e impacto, são priorizados aqueles riscos que, se ocorrerem, causarão o maior impacto ao projeto. A classificação dos riscos deve ter como base a figura 11.

Figura 11 : Matriz de Risco 1

Probabilidade	Impacto	Índice de Exposição ao Risco
25% (ocorrência baixa ou imperceptível)	1	0,25
50% (razoável)	3	1,5
75% (o risco é eminente)	5	3,75

Ou pela matriz da figura 12.

Figura 12: Matriz de Risco 2

	Imp. Baixo (10)	Imp. Médio (50)	Imp. Alto (100)
Prob. Alto (1,0)	Médio (10)	Alto (50)	Alto (100)
Prob. Médio (0,5)	Baixo (5)	Médio (25)	Alto (50)
Prob. Baixo (0,1)	Baixo (1)	Baixo (5)	Médio (10)

Sendo a escala da segunda matriz de Risco: Alto (>45 a 100), Médio (>5 a 45) e Baixo(1 a 5).

Há três estratégias para prevenir os riscos de um projeto, são ações que ampliam as oportunidades ou reduzem as ameaças sobre o projeto, estas que serão expostas abaixo:

a) **mitigar:** a mitigação procura reduzir a probabilidade e/ou a conseqüência de riscos adversos a limiares aceitáveis. Pode tomar a forma da implementação de um novo curso de ação, como a adoção de processos menos complexos, execução de um número maior de testes ou a escolha de um fornecedor mais estável. Ele pode envolver condições de mudanças em que a probabilidade de ocorrência de risco seja reduzida;

b) **evitar:** evitar o risco é mudar o plano de projeto para eliminar o risco ou a condição ou para proteger os objetivos do projeto destes impactos. Embora a equipe não possa eliminar todos os eventos de risco, alguns riscos específicos podem ser evitados. Alguns eventos de risco que surgem cedo no projeto podem ser evitados com requerimentos esclarecedores, obtendo-se informações, melhorando a comunicação ou consultando especialista. Reduzindo escopo para evitar atividades de alto risco, acrescentando recursos ou tempo, adotando uma abordagem familiar em vez de uma inovação, ou evitando um fornecedor desconhecido podem ser exemplos de evitar o risco;

c) **aceitar:** nesse caso aceitar o risco consiste em entender o risco e não fazer nada quanto a sua probabilidade e conseqüência. Simplesmente tomar o conhecimento sem adoção de medidas de controle.

8.1.2.3 Planejar Testes

Tem como objetivo direcionar e controlar as atividades de teste com base na avaliação de riscos utilizando de tais métodos:

a) **definir escopo:** consiste na definição dos requisitos que serão testados ou não tomando como base a análise dos riscos. A priorização dos requisitos é realizada a partir dos valores de exposição aos riscos;

b) **definir estratégias:** diversas estratégias podem ser utilizadas como base na análise dos riscos;
c) **definir cronograma:** também como base na análise dos riscos, os requisitos com prioridade alta serão planejados, projetados e executados nas primeiras iterações. Os requisitos com baixa prioridade só serão planejados, projetados e executados caso sobre tempo.

8.1.2.4 Projetar Testes

Tem como objetivo identificar e descrever os casos de teste para os requisitos e riscos a serem testado com base na análise de riscos. Tem como etapas:

a) **definir estratégia de caso de teste:** consiste na identificação dos cenários a serem testados, além da definição da abordagem e tipo de teste que serão utilizados. Esse processo sugere coberturas diferentes para os requisitos de acordo com suas importâncias. Por exemplo, requisitos com baixa prioridade serão testados somente os casos de testes relacionados aos riscos, caso haja tempo disponível. Já os requisitos com média prioridade serão testados os casos de testes relacionados aos riscos, fluxo principal e fluxos alterados, e por fim os requisitos com alta prioridade, serão testados os casos de teste relacionados aos riscos e todos os fluxos do requisito, o principal, exceção e alternativo;
b) **especificar projeto e procedimento de casos de teste:** os casos de teste que foram identificados são agora detalhados seguindo as estratégias definidas. Para cada risco identificado, testes são criados como propósito de mitigar os fatores de risco. O estilo de criação dos casos de teste para execução recomendado pelo processo é o independente, de forma que estes possam ser executados em qualquer ordem.

8.1.2.5 Executar Testes

Tem como objetivo executar testes e verificar a se o software está correto, avaliando os resultados e registrando os problemas encontrados.

a) **executar testes e reportar resultados:** a principal entrada para esta atividade é a planilha da execução criada pelo Projetista de Teste. Ao executar os testes, o testador gera os logs de resultados, como evidência das execuções e, quando falhas são encontradas, solicitações de mudanças são criadas.

8.1.2.6 Avaliar Testes

Sua finalidade é medir quantitativamente e qualitativamente o progresso dos testes e gerar um relatório de avaliação.

a) **avaliar resultado geral:** o projetista de testes avalia os resultados gerados na atividade Executar Testes e cria o relatório de resultado dos testes com número de casos de testes planejados, executados e seus resultados. Além disso, o projetista avalia as estratégias utilizadas e a cobertura dos casos de testes.

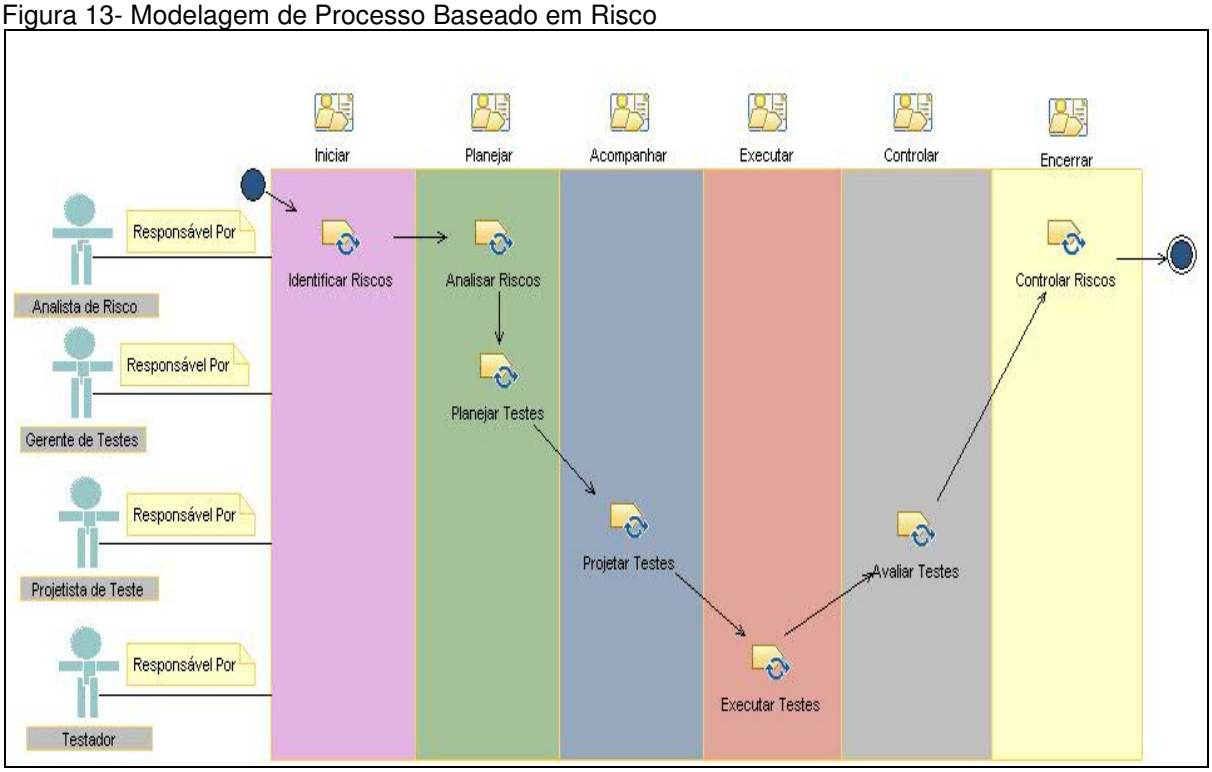
8.1.2.7 Controlar Riscos

Essa disciplina é responsável pelo acompanhamento dos riscos identificados.

a) **identificar riscos mitigados:** através do relatório de resultado dos testes e das solicitações de mudanças criadas, o Analista de Riscos identifica os riscos mitigados e atualiza o documento de riscos. Além disso, o analista gera um relatório informando todos os riscos mitigados ou o percentual de mitigação e uma nova lista de priorização dos riscos.

Pode-se observar o processo completo na figura 13, modelado na ferramenta EPF Composer.

Figura 13- Modelagem de Processo Baseado em Risco



Essas etapas que foram mostradas fazem parte do processo, que se cumpridas diminuíram os riscos de falhas no software, além de ter um menor custo para a empresa que fará utilidade do mesmo e mostrar que não é só essencial determinar processos de teste baseados em risco antes da implementação de um software, mas proporcionar às empresas a possibilidade de adotar processos de testes a serem tomados como base na hora do desenvolvimento do software para que não ocorram riscos inesperados e se tornem problemas maiores no futuro.

A seguir será feita uma validação do processo de teste baseado em risco utilizando as normas do modelo de processo MPS.BR.

8.2 APLICAÇÃO DAS NORMAS DE QUALIDADE MPS.BR

Para a aplicação das normas MPS.Br no processo de teste de software baseado em risco foi utilizados como base o nível de processo validação do nível D.

Para que o modelo de processo de teste de software baseado em risco apresentado seja válido de acordo com normas ele deve atender alguns critérios como:

8.2.1 VAL 1 - Os produtos de trabalho a serem validados são identificados

Na fase iniciar, que é primeira etapa do processo a ser executada, e a segunda etapa que diz respeito à fase de planejamento, são as mais relevantes, pois são nelas que serão identificados e analisados os riscos. Os riscos que são os principais itens a serem analisados no processo, para que não chegue ao usuário final contendo falhas.

8.2.2 VAL 2 - Uma estratégia de validação é desenvolvida e implementada, estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação

O processo foi criado e juntamente estabelecido papéis, fases e atividades, sendo que cada fase contém uma atividade e é executada por uma pessoa responsável.

Primeiramente na fase de iniciação a atividade de identificação de riscos tem como responsável o analista de riscos. Nessa fase são utilizadas como estratégias a elaboração de um questionário, a criação de uma lista de riscos, e também reuniões de Brainstorm.

Na fase de planejamento a pessoa responsável novamente é o analista de riscos e a atividade a qual executa é analisar riscos. Nessa fase que é utilizado o método para calcular a exposição ao risco e definir a prioridade dos riscos. Também nessa fase, para a atividade de planejar testes tem como responsável o gerente de teste da empresa, e na mesma são utilizados métodos como definir escopo, definir estratégias e por fim definir cronograma com base na análise de riscos.

Já a etapa de acompanhamento, o responsável é o projetista de teste e controla a atividade de projetar testes, essa na qual são definidas estratégias de caso de teste e também é especificado projeto e procedimentos de caso de testes.

O encarregado pela fase de execução é o testador e é responsável pela atividade de executar testes. Nessa atividade a estratégia é executar testes e reportar os resultados obtidos.

Na fase de controle o responsável pela atividade de avaliar os testes é o testador. Nessa atividade o objetivo é avaliar os resultados gerais alcançados.

Na fase de encerramento o responsável pela tarefa de controlar riscos é o analista de riscos e a estratégia é identificar os riscos mitigados.

8.2.3 VAL 3 - Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido

A partir das atividades do processo que identificam e analisam os riscos, são executadas as demais atividades essas que planejam os testes, projetam, executam, avaliam, e por ultimo a atividade de controle de risco. Nessas fases se utilizam de diversos critérios e procedimentos, como, identificar os cenários a serem testados, criação de planilhas, relatórios. Os ambientes e recursos que serão utilizados para o desenvolvimento das tarefas serão definidos a critério dos responsáveis pelas atividades citadas.

8.2.4 VAL 4 - Atividades de validação são executadas para garantir que o produto esteja pronto para uso no ambiente operacional pretendido

O processo em questão atende a esse critério da norma, pois contém as atividades de planejar testes, projetar, executar e ainda a de avaliar testes.

8.2.5 VAL 5 - Problemas são identificados e registrados

Nesse critério os riscos identificados no decorrer da execução do processo de teste serão identificados, analisados e controlados, para que possam ser tratados de maneira correta de acordo com suas prioridades, sendo que os mesmos poderão ser aceitos, mitigados, ou evitados de acordo com o analista de risco.

8.2.6 VAL 6 - Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas

Para esse critério são analisados os resultados obtidos, disponibilizados aos membros da equipe responsáveis e se necessário reiniciar todo o processo para novas averiguações.

8.2.7 VAL 7 - Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas

Nesse critério os riscos identificados já foram avaliados, sendo que já foram tomadas as providências necessárias, caso precisassem de alterações, as mesmas já foram feitas e toda analise e derivados também. Todas as etapas foram devidamente cumpridas para que os riscos existentes não prejudicassem o software e o mesmo esteja em perfeitas condições de chegar ao seu usuário final.

CONCLUSÃO

A tarefa de efetuar testes em software na maioria das empresas ainda é considerada secundária, como uma tarefa onde não se deve gastar muito com tempo e investimentos, sendo que o tempo reservado a testes de software, é em geral, menos da metade do tempo de desenvolvimento. Só que o custo ao entregar um produto sem realizar testes, muitas vezes pode ser muito mais elevado do que se o teste fosse executado antes da entrega final do produto.

Por isso a necessidade de sistematizar formas de evitar os custos elevadíssimos resultantes dos defeitos de software e dos erros.

Riscos são incertezas que possuem uma probabilidade de acontecerem e de causarem algum impacto, sendo esse positivo ou negativo para o projeto. O que não se pode descuidar é que se um risco ocorre, trará ao projeto impactos tais como custos, qualidade e tempo.

Com a competição entre as empresas de software esta havendo uma grande preocupação em aprimorar e aperfeiçoar os processos de testes em desenvolvimento de software com vistas a reduzir custos com manutenção e em produzir um produto de melhor qualidade.

Uma estratégia de testes bem executada é essencial para a garantia da qualidade de um software. Um software confiável e que implemente todos os requisitos especificados é o que os clientes esperam. Assim, o sucesso de um software depende diretamente de sua qualidade

Para garantir a qualidade nos produtos desenvolvidos muitas empresas brasileiras de desenvolvimento de software estão adotando o MPS.BR para apoiar as equipes de trabalho.

O presente trabalho apresenta os resultados da elaboração de um estudo de algumas normas de qualidade, atividades de teste de software e a modelagem de processo de teste de software.

Um grande motivo que levou ao desenvolvimento desta pesquisa foi a percepção em relação à grande deficiência no processo de teste de software hoje, evidenciada pela falta de uma estratégia bem definida para a realização dos mesmos, devido principalmente ao desconhecimento das técnicas e normas utilizadas para este tipo de atividade, sendo assim viu-se a necessidade de modelar

um processo de teste de software baseado em risco aliado as normas de qualidade do modelo MPS.BR utilizando o nível D para validar o processo de teste.

Outra observação conclusiva se refere ao fato de que a análise das normas estudadas comprova que mesmo sendo impossível testar um sistema por inteiro, a qualidade final dos softwares testados tende a melhorar significativamente se as suas sugestões forem seguidas, sendo que ter como base um processo de teste de software baseados em risco é de grande importância para realização de testes de uma empresa diminuindo a possibilidade de falhas humanas, inconsistência nos resultados e, principalmente, proporcionando maior rapidez na execução de tais tarefas.

Por fim, visto que esse trabalho não apresenta testes realizados, mas apenas um protótipo propõe-se como trabalhos futuros:

- Adequar a área de processo de verificação do nível D de maturidade do MPS.BR dentro do processo de teste baseado em risco.

- Testar e implementar o processo de testes dentro de uma empresa;

- Adequar o processo de teste em outras normas de qualidade e fazer uma análise entre os resultados obtidos.

REFERÊNCIAS

ABNT- ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE. **NBR ISO 9001: Sistemas de gestão da qualidade- Requisitos.** Rio de Janeiro, 2000.

ABNT NBR ISO/IEC 15504-2:2008. **Tecnologia da informação -Avaliação de processo Parte 2: Realização de uma avaliação.**

AMLAND, S. **Risk Based Testing and Metrics:** ***Risk Analysis fundamentals and metrics for software testing including a financial application case study.*** In: 5o International Conference EuroSTAR'99, 1999.

BACH, J. James Bach. **Risk-Based Testing:** ***How to conduct heuristic risk analysis.*** In: Software Testing & Quality Engineering Magazine, 1999. Tradução livre do autor desta Monografia. Disponível em: <http://www.satisfice.com/articles/hrbt.pdf>. Acesso em: 15 maio 2011.

BASTOS, Aderson; RIOS, Emerson; CRISTALLI, Ricardo; MOREIRA, Trayahú. **Base de Conhecimento em Teste de Software**. 2a Edição. São Paulo: Martins, 2007, 263p.

BELLOQUIM, A. **CMMI: O futuro do CMM.** Congresso Fenasoft 2001. InfoChoose Technologies, Nº 42 Ano III, fevereiro/2002. Disponível em: < http://www.choose.com.br/infochoose/infochoose42.htm#1>
Acesso em 20 de set de 2011.

BUENO, Cassiane de Fátima Dos Santos; CAMPELO, Gustavo Bueno.**Qualidade de Software.** Recife: 2011. 28 p. Disponível em: <http://www.cin.ufpe.br/~mrsj/Qualidade/Qualidade%20de%20Software.pdf>. Acesso em: 21 out. 2011.

CÔRTES, M. L.; CHIOSSI, Thelma C. dos Santos. **Modelos de Qualidade de Software.** 1. ed. Campinas: Editora da Unicamp, 2001.

DELAMARO, Marcio Eduardo,. MALDONADO, José Carlos,. JINO, Mario. **Introdução ao Teste de Software**. Rio de Janeiro, 2007. 394p.

DEUTSCH, M. Verification and Validation. In: PRESSMAN, Roger. **Engenharia de Software**. São Paulo: Makron Books, 1995.

FERREIRA, Wilker Felix. **MPS.BR um estudo do modelo MPS.BR como benefício para pequenas e médias empresas**. 2009. 69 f. Tcc (Graduação) - Universidade Estadual de Goiás, Goiás, 2009. Disponível em: <http://www.softwarepublico.gov.br/file/17234990/MONOGRAFIA_WILKER_MPS.BR.pdf>. Acesso em: 11 maio 2011.

GOMES, Nelma da Silva. **QUALIDADE DE SOFTWARE – UMA NECESSIDADE.** 2000. 13 f. Monografia (Especialização) - Curso de Gestão Estratégica da Informação, Ucp, Petrópolis, 2000.

ISO/IEC, 2003, INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL COMISSION. **ISO/IEC 15504: Information Technology – Process Assessment. Part 1 – Concepts and vocabulary; part 2 – Performing an assessment; part 3 – Guidance on performing an assessment; part 4 – Guidance on use for process improvement and process capability determination; and part 5 – An exemplar process assessment model.**

KANER, C.; FALK, J.; NGUYEN, H. Q. **Testing computer software**. 2a edição, Wiley, 1999.

KOSCIANSKI, Andre; SOARES, Michel Dos Santos. **Qualidade de Software**. 2ª edição. São Paulo: Novatec, 2007. 400 p.

MACHADO, Cristina Ângela Filipak. **Definindo processos do ciclo de vida de software usando a norma NBR ISO/IEC 12207 e suas ementas 1 e 2.** Lavras: UFLA/FAEPE, 2006.

MCMAHON, Keith; Risk Based Testing, ST Labs, WA, 1998 apud ROSENBERG, Linda H.; STAPKO, Ruth; GALLO, Albert ; **Risk-based Object Oriented Testing;** NASA SEW24 - Twenty-Fourth Annual Software Engineering Workshop. Tradução livre do autor desta Monografia,1999.

MCT, **Ministério da Ciência e Tecnologia.** 2008. Disponível em: <www.mct.gov.br>. Acesso em: 20 de Set de 2011.

MERTINS, C. F. **Desenvolvimento e Gestão de Requisitos de Software**. Monografia (Curso de Bacharel em Informática) – Faculdade de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2004.

MIT; **Slicing the Knowledge-based Economy in Brazil, China and India: a tale of 3 software industries**. 2003.

MOLINARI, Leonardo. **Testes Funcionais de Software.** Santa Catarina: Visual Books, 2008. 214p.

MPS.BR, 2007a – Associação para Promoção da Excelência do Software Brasileiro – SOFTEX. **MPS.BR – Guia Geral**, versão 1.2**,** junho 2007. Disponível em: <www.softex.br>. Acesso em 22 de Set de 2011.

MPS.BR, 2007b – Associação para Promoção da Excelência do Software Brasileiro – SOFTEX. **MPS.BR – Guia de Avaliação**, versão 1.1**,** junho 2007. Disponível em: <www.softex.br>. Acesso em 22 de Set de 2011.

MPS.BR, 2007c – Associação para Promoção da Excelência do Software Brasileiro – SOFTEX. **MPS.BR – Guia de Aquisição**, versão 1.2, junho 2007**.** Disponível em: <www.softex.br>. Acesso em 22 de Set de 2011.

MYERS, Glenford J. **The Art of Software Testing, Ed. 2.** New York: John Wiley & Sons, Nova Jérsei: 2004, 177p.

PALZA, Edgardo, FUHRMAN, Christopher P., ABRAN, Alain: **Establishing a Generic and Multidimensional Measurement Repository in CMMI context.** Annual NASA Goddard Software Engineering Workshop in SEW, 2003.

PÊSSOA, Marcelo Schneck de Paula. **Modelo Integrado de Maturidade da Capacidade de Processo**. Lavras: UFLA/FAEPE, 2005.

PMI, Project Management Institute. Um **Guia do Conjunto de Conhecimento de Projetos – Guia PMBOK.** PMI Standards Comitee, 2004

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Pearson Makron Books, 2007. 1056p.

PRESSMAN, Roger S.. **Engenharia de Software.**6. ed. Rio de Janeiro: Mcgraw-hill do Brasil, 2008.

PROJECT MANAGEMENT INSTITUTE (PMI). A **Guide to the Project Management Body of Knowledge: *PMBOK guide***. 3. ed. Pennsylvania, USA, 2004, 388 p.

RINCON, André Mesquita. **Qualidade de Software. In: XI Encontro de Estudantes de Informática do Tocantins, 2009, Palmas**. Anais do XI Encontro de Estudantes de Informática do Tocantins. Palmas: Centro Universitário Luterano de Palmas, 2009. p. 75-86. Disponível em: http://tinyurl.com/yj8f4kh.

RIOS, Emerson. **Análise de Riscos em Projetos de Teste de Software**. Rio de Janeiro: Alta Books, 2007, 131p.

RIOS, Emerson; MOREIRA FILHO, Trayahú R. . **Teste de software**. 2. ed. rev. e ampl. Rio de Janeiro: Alta Books, 2006. 222p.

SEI - SOFTWARE ENGINEERING INSTITUTE. **CMMI for Development (CMMI-DEV), Version 1.2, Technical Report CMU/SEI-2006-TR-008.** Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 2006. Disponível em: http://www.sei.cmu.edu/reports/06tr008.pdf. Acesso em 10 outubro de 2011.

SEI, 2009a. Software Engineering Istitute – Site Oficial. Disponível em <http://www.sei.cmu.edu/>. Acesso em 20 de setembro de 2011.

SEI, 2009b. Software Engineering Istitute – CMMI for Development (CMMIDEV). Disponível em <http://www.sei.cmu.edu/cmmi/tools/dev/>. Acesso em 20 de setembro de 2011.

SOFTEX – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – **SOFTEX. MPS.BR – Guia de Aquisição: 2007,** junho 2007. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2007.pdf. Acesso em: 26 set. 2011

SOFTEX, 2009a – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – **SOFTEX. MPS.BR – Guia Geral: 2009,** maio 2009. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf. Acesso em: 26 set. 2011.

SOFTEX, 2009b – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – **SOFTEX. MPS.BR – Guia de Avaliação: 2009,** maio 2009. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf. Acesso em: 26 set. 2011.

SOFTEX, 2009c – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – **SOFTEX. MPS.BR – Guia de Aquisição: 2009,** maio 2009. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf. Acesso em: 26 set. 2011.

SOUZA, Ellen; GUSMÃO, Cristine; ROCHA, Humberto. **RBTProcess - Proposta de Modelo de Processo de Teste de Software baseado em Risco.** Recife, 2008. 6 p.

REFERÊNCIAS COMPLEMENTARES

ABNT NBR ISO/IEC 15504 – Partes 1 a 4: **Tecnologia da Informação – Avaliação de Processo.** Rio de Janeiro, 2008.

CARDOSO, Fernando Schiavo. **TESTE DE SOFTWARE BASEADO EM RISCO.** 2008. 43 f. Monografia (Graduação) - Faculdade de Jaguariúna, Jaguariúna, 2008. Disponível em: <http://bibdig.poliseducacional.com.br/document/?down=169>. Acesso em: 17 maio 2011.

GERRARD, Paul; **Risk-Based E-Business Testing-Part 1, Risk and Test Estrategy.** Disponível em: <http://www.p2080.co.il/go/p2080h/files/3266858756.pdf >. Acesso em: 15 maio de 2011.

GUERRA, Ana Cervigni; COLOMBO, Regina Maria Thienne. . **Tecnologia da informação:** qualidade de produto de software. Brasília: Ministério da Ciência e Tecnologia, 2009. 429 p.

INTHURN, Cândida. **Qualidade & teste de software: engenharia de software, qualidade de software, qualidade de produtos de software, teste de software, formalização do processo de teste, aplicação prática dos testes.** Florianópolis: Visual Books, 2001. 108 p. ISBN 8575020269.

MOLINARI, Leonardo. **Testes de Software**. São Paulo: Érica, 2003. 232 p.

MORAES, Edson Andrade de. **Teste de Software Baseado em Risco**. 2006. 25 f. Monografia (Graduação) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.dbd.pucrio.br/depto_informatica/06_07_moraes.pdf>. Acesso em: 17 maio 2011.

NOGUEIRA, Marcelo. **Engenharia de Software.** Rio de Janeiro: Ciência Moderna, 2009. 224 p.

OLIVEIRA, Ivan Jose De Mecenas Silva & Vivianne De. **Qualidade em Software.** Rio de Janeiro: Alta Books, 2005. 158 p.

PAULA FILHO, Wilson De Padua. **Engenharia de Software Fundamentos, Métodos e Padrões.** 3. ed. Rio de Janeiro: Ltc, 2009. 1256 p.

PEZZÈ, Mauro; YOUNG, Michal. . **Teste e análise de software: processo, princípios e técnicas**. Porto Alegre: Bookman, 2008. 512 p.

PFLEEGER, Shari L. **Engenharia de Software**: Teoria na Prática. 2. ed. São Paulo: Prentice Hall, 2004.

RODRIGUES, Juliana França. **Avaliação da Implantação do MPS.BR: Um estudo empírico sobre benefícios, dificuldades e fatores de sucesso.** 2009. 191 f. Dissertação (Mestrado) - Universidade Metodista De Piracicaba, Piracicaba, 2009. Disponível em: <http://www.unimep.br/phpg/bibdig/pdfs/2006/PYOBHYRIFGCF.pdf>. Acesso em: 27 maio 2011.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. . **Qualidade de software:** teoria e prática. São Paulo: Prentice Hall, 2001. 303 p. ISBN 8587918540.

SOFTEX – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – **SOFTEX. MPS.BR – Guia de Aquisição: 2011,** junho 2011. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2011.pdf. Acesso em: 26 set. 2011.

SOMMERVILLE, Ian. **Engenharia de Software.** 8ª ed. São Paulo: Pearson Addison-wesley, 2007.

SOUZA, Ellen Polliana Ramos. **RBTProcess: Modelo de Processo de Teste de Software baseado em Riscos.** 2007. 12 f. Dissertação (Mestrado) - Upe, Recife, 2008. Disponível em: <http://www.lbd.dcc.ufmg.br:8080/colecoes/wtdes/2008/002.pdf>. Acesso em: 15 de maio 2011.

WOODS, Peter; BEALE, David. . **Os benefícios da análise baseada em risco durante a validação de sistemas**. Pharmaceutical Technology (Ed. Brasileira), São Paulo, v.13, n.6, 2009.

APÊNDICE

Teste de Software Baseado em Risco Aplicado em Normas de Qualidade

Denise Tomasi Isoppo[1], Ana Cláudia Garcia Barbosa[2]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – 88806-000– Criciúma – SC – Brasil

[2]Departamento de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – 88806-000– Criciúma – SC – Brasil

denise_tomasi@hotmail.com, agb@unesc.net

Abstract. *This article research shows as a main subject software testing based in a risk that consists in a group of activities that favors the identification of risk factors associated to the software, and had as an objective the modeling of a software testing process based in risks, the same appropriated to the quality standards. This project shows a larger modeling than the ones found in the literature, because its consists in activities, artifacts, roles that has as an objective to guide the testing engineers the use in this approach based in risks, as a form to promote a better use from the resources and the time. To meet the needs from the field the process was appropriated to the quality standards using the Brazilian software process modeling.*

Resumo. *Este artigo apresenta como tema central teste de software baseado em risco que consiste num conjunto de atividades que favorecem a identificação de fatores de risco associados ao software, e teve como objetivo a modelagem de um processo de teste de software baseado em risco adequado às normas de qualidade. Esse projeto mostra um modelo mais amplo do que os encontrados na literatura, pois consiste em atividades, artefatos, papéis que tem como finalidade guiar os engenheiros de teste na utilização dessa abordagem baseada em risco, de forma a promover melhor uso de recursos e tempo. Para atender as necessidades do mercado o processo foi adequado às normas de qualidade, utilizando o modelo de processo de software brasileiro.*

1. Introdução

Produtos de qualidade, de fácil uso, fácil manutenção são o que todos procuram no mercado, devido a este fato é que hoje a maioria das empresas está aderindo às normas de qualidade.

O desenvolvimento de produtos de software está em constante evolução, e as empresas desenvolvedoras vêm enfrentando problemas quanto ao usuário final do software, a cada dia aumenta o nível de exigência por parte dos clientes no que diz respeito ao controle de qualidade do produto, no desempenho, complexidade e nos efeitos que o seu uso possa produzir na organização.

As falhas de sistema que chegam ao usuário, por exemplo, são fatores comuns nas companhias e os mais preocupantes. Apesar dos testes serem atividades difíceis e de alto custo para ser realizado em uma empresa devido aos inúmeros erros possíveis, os mesmos se tornaram parte essencial no desenvolvimento de um software. Além das possíveis falhas, os

eventuais problemas afetados por riscos imprevistos que ocorrem comumente em projetos, aqueles não planejados ou simplesmente os ignorados, sendo que com o aumento da complexidade e o crescimento do projeto os mesmos se agravam cada vez mais, tornando cada vez piores de serem resolvidos.

Se a área de teste não estiver bem organizada os defeitos vão persistir a ocorrer num estágio onde os custos são cada vez maiores.

Com intuito de minimizar problemas como estes as empresas estão buscando se aprimorar mais nas atividades teste de software, sendo que os mesmos passaram a ser executados por equipes especializadas e as empresas criaram áreas na sua estrutura organizacional para poder cumprir esse papel.

O teste de software baseado em risco consiste em identificar, priorizar e criar casos de teste que exploram os riscos no sistema, e tem se mostrado de grande utilidade para as empresas, aproveitando melhor os recursos e o tempo do teste, perfazendo um processo que aborde as áreas de maior risco do sistema e garantindo que o produto final não saía errado.

Com base neste contexto, esta pesquisa propõe definir um processo de caso de teste baseado em risco, visando a aplicação das normas de qualidade MPS.BR.

2. Metodologia

Apesar dos testes serem atividades difíceis e de alto custo para ser realizado em uma empresa devido aos inúmeros erros possíveis, os mesmos se tornaram parte essencial no desenvolvimento de um software. Se a área de teste não estiver bem organizada os defeitos vão persistir a ocorrer num estágio onde os custos são cada vez maiores.

Com intuito de minimizar problemas como estes esse projeto busca demonstrar um modelo de processo que facilita a execução da etapa de testes de software de uma empresa, sendo que os mesmos passam a ser executados por equipes especializadas.

O presente trabalho tem a finalidade de modelar um processo de teste de software aplicado juntamente a normas de qualidade MPS.BR, que é um modelo de processo de software brasileiro voltada para realidade do mercado brasileiro e também às pequenas e médias empresas, mais especificamente no nível D, este que engloba a parte de validação a qual será avaliada no processo para saber se o mesmo está ou não apropriado as normas exigidas.

O desenvolvimento do projeto consiste no estudo de modelos de qualidade de software, no estudo da ferramenta EPF Composer, utilizada para criação do processo, no desenvolvimento dos processos de teste baseados em risco e por fim a análise e os resultados que foram obtidos do ínicio ao fim do projeto.

Primeiramente foi realizado um levantamento bibliográfico a fim de se obter referencial teórico, em seguida voltou-se o estudo a normas e qualidade de software. A segunda etapa foi criação dos processos de teste baseados em risco, para isso foram realizadas diversas pesquisas sobre testes de software, para poder entender e aplicar seus conceito, e também posteriormente poder aplicar normas de qualidade.

Dentre todas essas abordagens encontradas na literatura a mais completa que fornece guias, métricas, artefatos, papéis e atividades é o RBTProcess, processos de teste de software baseados em risco.

Esse processo é formado a partir da identificação das principais atividades presentes no gerenciamento de riscos e no processo de teste de software padrão, as atividades de

software também sofreram mudanças com relação ao padrão, pois agora passam a ser guiadas por risco.

As fases dessa abordagem podem ser observadas na figura 1.

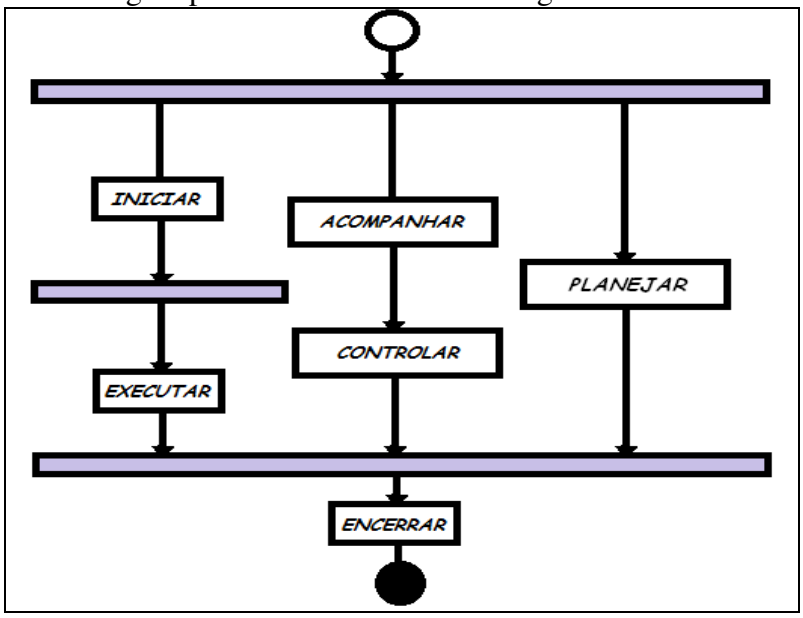


Figura 1. Diagrama de Atividade

Os processos de teste baseados em risco, como mostra no diagrama possuem fases distintas, cada uma dessas fases desempenha uma atividade que é controlada e de extrema responsabilidade de um ator, ou seja, é realizada por um membro da equipe de testes.

Na primeira, fase iniciar, é necessário que sejam identificadas as necessidades e levantadas as informações e estimativas para a elaboração de um plano que contenha os principais objetivos e atividades que serão executados.

A fase de planejamento tem como foco a elaboração inicial dos testes com base na análise dos riscos. É nessa fase que serão elaboradas estratégias e o plano de teste, como também será descrito um caminho a ser seguido nas próximas etapas do processo, além de estabelecer o que vai ser testado, quando serão executados e em quanto tempo. Essa fase deve ser mantida ativa até que o projeto seja concluído.

A fase seguinte, de acompanhamento, tem como objetivo acompanhar todo processo para detecção de possíveis defeitos e falhas, acompanha o processo desde o início até o fim.

Os resultados da execução dos testes são coletados e controlados na fase de controle. Compreende a atividade de avaliar testes onde são verificados números, estratégias, tempo de execução, solicitação de mudanças dos testes.

Na fase executar, os casos de testes planejados e projetados são executados. Abrange a atividade executar testes. O marco desta fase é a execução de casos de testes que verificam a existência ou não dos riscos identificados.

Por último na fase de encerramento compreende a atividade controlar risco, onde o acompanhamento dos riscos é realizado e tem como objetivo obter a aceitação final, avaliar e relatar as lições aprendidas na execução do projeto, gerar estimativa e garantir a qualidade do produto. Seu cumprimento é necessário para que o software possa ser liberado com uma margem segura e comprovada através de indicadores que a qualidade e a ausência de defeitos críticos estejam totalmente corrigidas.

Cada fase do processo necessita de um membro, uma pessoa da equipe de teste, que tem um papel fundamental na execução de uma atividade.

2.1 Papéis executados na abordagem de teste de software baseados em risco

Fazem parte da equipe de teste o analista de risco, gerente de risco, projetista de teste e testador, com exceção do analista de risco que é um papel da gerencia de risco e essencial para identificação, análise e controle dos riscos, os outros papéis executados são os mesmos de um processo de teste padrão.

O analista é um membro da equipe de testes, um engenheiro de testes, este possui conhecimento sobre teste de software, identificação e análise dos riscos, além de profundo conhecimento sobre os requisitos do software.

Já para o gerente de testes são necessários alguns conhecimentos no que se refere à processos de testes, análise e planejamento e acompanhamento.

O projetista de testes necessita de um conhecimento aprofundado sobre os requisitos do sistema e tecnologia adotada, sobre os riscos, ferramentas, técnicas para construção, estratégias e tipos de testes.

O testador carece de conhecimento sobre o processo de teste, ferramentas utilizadas para execução, configuração do ambiente de execução e solicitação de mudança, além do conhecimento sobre os requisitos do sistema.

A seguir será especificada cada atividade desempenhada.

2.2 Atividades realizadas na abordagem de teste de software baseados em risco

As atividades têm como objetivo definir como os testes serão conduzidos no projeto.

2.2.1 Identificar Riscos

Essa atividade é proveniente do gerenciamento de risco e tem como objetivo identificar os riscos técnicos associados aos requisitos do sistema que podem de alguma forma afetar o sucesso do projeto e que também podem ser mitigados através da execução de testes e documentar suas características.

De acordo [PMI 2004] no PMBOK essa atividade pode ser dividida em três categorias que auxilia na identificação dos riscos que será exposta a seguir:

a) **revisar fontes e categoria de riscos:** nessa etapa o objetivo é avaliar e adequar as listas de riscos e o questionário para o software que será testado;
b) **responder questionário:** é a primeira etapa para a identificação dos riscos. Para casa requisito, são feitas algumas perguntas e respondidas por participantes e que de acordo com a resposta, é indicada a existência do risco ou não;
c) **realizar reunião de Brainstorm:** tem como objetivo validar os riscos levantados na atividade anterior. Para guiar a reunião são utilizadas as listas de risco. As reuniões de Brainstorm com listas de riscos, visam tornar a reunião mais eficiente. Como produto dessa atividade tem-se uma lista de riscos identificados para cada requisito.

2.2.2 Analisar Riscos

Tem como finalidade a priorização dos requisitos a serem testados com base na análise dos riscos técnicos identificados, mede a probabilidade e as conseqüências dos riscos e estima

suas implicações para os objetivos do projeto. É nessa etapa que é calculado a exposição do risco e analisado.

2.2.2.1 Calcular a exposição do risco

Para se fazer uma análise de risco é necessária uma avaliação de alguns fatores, como a probabilidade da ocorrência do risco e o impacto de haver uma perda ligada ao risco. A fórmula abaixo foi proposta por [Rios 2007].

R = Pb x I **(1)**

Sendo R a exposição ao risco, Pb a probabilidade de ocorrência do risco vezes I impacto do risco.

Para que se possa evitar, mitigar ou aceitar um risco é necessário que o mesmo seja calculado utilizando a fórmula, para se saber o qual e quanto esse risco deve ser analisado.

Os valores que serão utilizados para calcular são propostos pela própria autora do trabalho e serão expostos em uma matriz de risco.

O impacto pode ser classificado de acordo com a gravidade sendo: baixa, o impacto é irrelevante para o projeto podendo ser resolvido facilmente; média: impacto é relevante para o projeto e necessita de um gerenciamento mais preciso e pode prejudicar o resultado do projeto; alta: impacto extremamente elevado e os resultados poderão ser comprometidos.

Por meio da matriz de probabilidade de ocorrência e impacto, são priorizados aqueles riscos que, se ocorrerem, causarão o maior impacto ao projeto. A classificação dos riscos deve ter como base a figura 2.

Probabilidade	Impacto	Índice de Exposição ao Risco
25% (ocorrência baixa)	1	0,25
50% (razoável)	3	1,5
75% (o risco é eminente)	5	3,75

Figura 2. Matriz de Risco 1

Há três estratégias para prevenir os riscos de um projeto, são ações que ampliam as oportunidades ou reduzem as ameaças sobre o projeto, estas que serão expostas abaixo:

a) **mitigar:** a mitigação procura reduzir a probabilidade e/ou a conseqüência de riscos adversos a limiares aceitáveis. Pode tomar a forma da implementação de um novo curso de ação.

b) **evitar:** evitar o risco é mudar o plano de projeto para eliminar o risco ou a condição ou para proteger os objetivos do projeto destes impactos.

c) **aceitar:** nesse caso aceitar o risco consiste em entender o risco e não fazer nada quanto a sua probabilidade e conseqüência. Simplesmente tomar o conhecimento sem adoção de medidas de controle.

2.2.3 Planejar Testes

Tem como objetivo direcionar e controlar as atividades de teste com base na avaliação de riscos utilizando de tais métodos:

a) **definir escopo:** consiste na definição dos requisitos que serão testados ou não tomando como base a análise dos riscos. A priorização dos requisitos é realizada a partir dos valores de exposição aos riscos;
b) **definir estratégias:** diversas estratégias podem ser utilizadas como base na análise dos riscos;
c) **definir cronograma:** também como base na análise dos riscos, os requisitos com prioridade alta serão planejados, projetados e executados nas primeiras iterações. Os requisitos com baixa prioridade só serão planejados, projetados e executados caso sobre tempo.

2.2.4 Projetar Testes

Tem como objetivo identificar e descrever os casos de teste para os requisitos e riscos a serem testado com base na análise de riscos. Tem como etapas:

a) **definir estratégia de caso de teste**: consiste na identificação dos cenários a serem testados, além da definição da abordagem e tipo de teste que serão utilizados. Esse processo sugere coberturas diferentes para os requisitos de acordo com suas prioridades.
b) **especificar projeto e procedimento de casos de teste:** os casos de teste que foram identificados são agora detalhados seguindo as estratégias definidas. Para cada risco identificado, testes são criados como propósito de mitigar os fatores de risco. O estilo de criação dos casos de teste para execução recomendado pelo processo é o independente, de forma que estes possam ser executados em qualquer ordem.

2.2.5 Executar Testes

Tem como objetivo executar testes e verificar a se o software está correto, avaliando os resultados e registrando os problemas encontrados.

a) **executar testes e reportar resultados:** a principal entrada para esta atividade é a planilha da execução criada pelo Projetista de Teste. Quando falhas são encontradas, solicitações de mudanças são criadas.

2.2.6 Avaliar Testes

Sua finalidade é medir quantitativamente e qualitativamente o progresso dos testes e gerar um relatório de avaliação.

a) **avaliar resultado geral:** o projetista de testes avalia os resultados gerados na atividade Executar Testes e cria o relatório de resultado dos testes com número de casos de testes planejados, executados e seus resultados.

2.2.7 Controlar Riscos

Essa disciplina é responsável pelo acompanhamento dos riscos identificados.

a) **identificar riscos mitigados:** através do relatório de resultado dos testes e das solicitações de mudanças criadas, o Analista de Riscos identifica os riscos mitigados e atualiza o documento de riscos. Além disso, o analista gera um relatório informando todos os riscos mitigados ou o percentual de mitigação e uma nova lista de priorização dos riscos.

Pode-se observar o processo completo na figura 3, modelado na ferramenta EPF Composer.

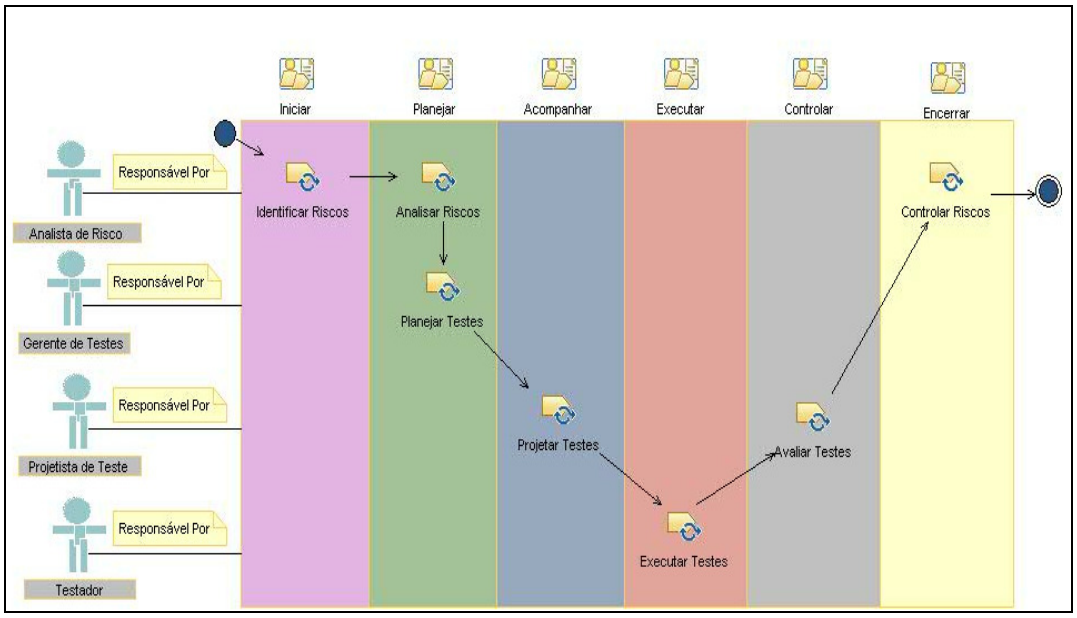


Figura 3. Modelagem de Processo Baseado em Risco

Essas etapas que foram mostradas fazem parte do processo, que se cumpridas diminuíram os riscos de falhas no software, além de ter um menor custo para a empresa que fará utilidade do mesmo e mostrar que não é só essencial determinar processos de teste baseados em risco antes da implementação de um software, mas proporcionar às empresas a possibilidade de adotar processos de testes a serem tomados como base na hora do desenvolvimento do software para que não ocorram riscos inesperados e se tornem problemas maiores no futuro.

A seguir será feita uma validação do processo de teste baseado em risco utilizando as normas do modelo de processo MPS.BR.

3. Resultados

Para a aplicação das normas MPS.BR no processo de teste de software baseado em risco foi utilizados como base o nível de processo validação do nível D.

Para que o modelo de processo de teste de software baseado em risco apresentado seja válido de acordo com normas ele deve atender alguns critérios como:

3.2.1 VAL 1 - Os produtos de trabalho a serem validados são identificados

Na fase iniciar, que é primeira etapa do processo a ser executada, e a segunda etapa que diz respeito à fase de planejamento, são as mais relevantes, pois são nelas que serão identificados e analisados os riscos. Os riscos que são os principais itens a serem analisados no processo, para que não chegue ao usuário final contendo falhas.

3.2.2 VAL 2 - Uma estratégia de validação é desenvolvida e implementada, estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação

Primeiramente na fase de iniciação a atividade de identificação de riscos tem como responsável o analista de riscos. Nessa fase são utilizadas como estratégias a elaboração de um questionário, a criação de uma lista de riscos, e também reuniões de Brainstorm.

Na fase de planejamento a pessoa responsável novamente é o analista de riscos e a atividade a qual executa é analisar riscos. Nessa fase que é utilizado o método para calcular a exposição ao risco e definir a prioridade dos riscos. Também nessa fase, para a atividade de planejar testes tem como responsável o gerente de teste da empresa, e na mesma são utilizados métodos como definir escopo, definir estratégias e por fim definir cronograma com base na análise de riscos.

Já a etapa de acompanhamento, o responsável é o projetista de teste e controla a atividade de projetar testes, essa na qual são definidas estratégias de caso de teste e também é especificado projeto e procedimentos de caso de testes.

O encarregado pela fase de execução é o testador e é responsável pela atividade de executar testes. Nessa atividade a estratégia é executar testes e reportar os resultados obtidos.

Na fase de controle o responsável pela atividade de avaliar os testes é o testador. Nessa atividade o objetivo é avaliar os resultados gerais alcançados.

Na fase de encerramento o responsável pela tarefa de controlar riscos é o analista de riscos e a estratégia é identificar os riscos mitigados.

3.2.3 VAL 3 - Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido

A partir das atividades do processo que identificam e analisam os riscos, são executadas as demais atividades essas que planejam os testes, projetam, executam, avaliam, e por ultimo a atividade de controle de risco. Nessas fases se utilizam de diversos critérios e procedimentos, como, identificar os cenários a serem testados, criação de planilhas, relatórios. Os ambientes e recursos que serão utilizados para o desenvolvimento das tarefas serão definidos a critério dos responsáveis pelas atividades citadas.

3.2.4 VAL 4 - Atividades de validação são executadas para garantir que o produto esteja pronto para uso no ambiente operacional pretendido

O processo em questão atende a esse critério da norma, pois contém as atividades de planejar testes, projetar, executar e ainda a de avaliar testes.

3.2.5 VAL 5 - Problemas são identificados e registrados

Nesse critério os riscos identificados no decorrer da execução do processo de teste serão identificados, analisados e controlados, para que possam ser tratados de maneira correta de acordo com suas prioridades, sendo que os mesmos poderão ser aceitos, mitigados, ou evitados de acordo com o analista de risco.

3.2.6 VAL 6 - Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas

Para esse critério são analisados os resultados obtidos, disponibilizados aos membros da equipe responsáveis e se necessário reiniciar todo o processo para novas averiguações.

3.2.7 VAL 7 - Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas

Nesse critério os riscos identificados já foram avaliados, sendo que já foram tomadas as providências necessárias, caso precisassem de alterações, as mesmas já foram feitas e toda analise e derivados também. Todas as etapas foram devidamente cumpridas para que os riscos

existentes não prejudicassem o software e o mesmo esteja em perfeitas condições de chegar ao seu usuário final.

4. Conclusão

A tarefa de efetuar testes em software na maioria das empresas ainda é considerada secundária, como uma tarefa onde não se deve gastar muito com tempo e investimentos, sendo que o tempo reservado a testes de software, é em geral, menos da metade do tempo de desenvolvimento. Só que o custo ao entregar um produto sem realizar testes, muitas vezes pode ser muito mais elevado do que se o teste fosse executado antes da entrega final do produto.

Por isso a necessidade de sistematizar formas de evitar os custos elevadíssimos resultantes dos defeitos de software e dos erros.

Uma estratégia de testes bem executada é essencial para a garantia da qualidade de um software. Um software confiável e que implemente todos os requisitos especificados é o que os clientes esperam. Assim, o sucesso de um software depende diretamente de sua qualidade.

O presente trabalho apresenta os resultados da elaboração de um estudo de algumas normas de qualidade, atividades de teste de software e a modelagem de processo de teste de software.

Um grande motivo que levou ao desenvolvimento desta pesquisa foi a percepção em relação à grande deficiência no processo de teste de software hoje, evidenciada pela falta de uma estratégia bem definida para a realização dos mesmos, devido principalmente ao desconhecimento das técnicas e normas utilizadas para este tipo de atividade, sendo assim viu-se a necessidade de modelar um processo de teste de software baseado em risco aliado as normas de qualidade do modelo MPS.BR utilizando o nível D para validar o processo de teste.

Outra observação conclusiva se refere ao fato de que a análise das normas estudadas comprova que mesmo sendo impossível testar um sistema por inteiro, a qualidade final dos softwares testados tende a melhorar significativamente se as suas sugestões forem seguidas, sendo que ter como base um processo de teste de software baseados em risco é de grande importância para realização de testes de uma empresa diminuindo a possibilidade de falhas humanas, inconsistência nos resultados e, principalmente, proporcionando maior rapidez na execução de tais tarefas.

Referências

Ferreira, Wilker Felix. (2009) “MPS.BR um estudo do modelo MPS.BR como benefício para pequenas e médias empresas”. 2009. 69 f. Tcc (Graduação) - Universidade Estadual de Goiás, Goiás, 2009. Disponível em: <http://www.softwarepublico.gov.br/file/17234990/MONOGRAFIA_WILKER_MPS.BR.pdf>. Acesso em: 11 maio 2011.

Moraes, Edson Andrade de. (2006) “Teste de Software Baseado em Risco”. 25 f. Monografia (Graduação) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.dbd.pucrio.br/depto_informatica/06_07_moraes.pdf>. Acesso em: 17 maio 2011.

PMI, Project Management Institute. (2004) “Um Guia do Conjunto de Conhecimento de Projetos – Guia PMBOK”. PMI Standards Comitee.

Pressman, Roger S.. (2008) “Engenharia de Software”. 6. ed. Rio de Janeiro: Mcgraw-hill do Brasil.

Rios, Emerson. (2007) “Análise de Riscos em Projetos de Teste de Software”. Rio de Janeiro: Alta Books, 131p.

Souza, Ellen; Gusmão, Cristine; Rocha, Humberto. (2008) “RBTProcess - Proposta de Modelo de Processo de Teste de Software baseado em Risco”. Recife, 6 p.