CERTIFICAÇÃO E QUALIDADE DE SOFTWARE

Professor: Paulo Cesar Azevedo Teixeira.

Conteúdo Programático:

Unidade 1. Introdução e Conceitos Básicos.
Unidade 2. Qualidade de Software
Unidade 3. Processo de Controle da Qualidade
Unidade 4. Perfil Profissional de Qualidade
Unidade 5. CMM
Unidade 6. PSP
Unidade 7. SPICE
Unidade 8. Medidas da Qualidade de Software

Bibliografia:

- Pressman, Roger S. - Engenharia de Software - 5.ed. - Rio de Janeiro: McGraw-Hill, 2002.
- Weinberg, G. M. - Software com Qualidade - 1.ed. - São Paulo: Brasport, 1999.
- Rezende, D. A. – Engenharia de Software e Sistemas de Informação – 2.ed. – Rio de Janeiro: Brasport, 2002.

Unidade 1 – Introdução e Conceitos Básicos

1.1 – QUAL O MELHOR CONCEITO PARA QUALIDADE?

A Qualidade, nos dias atuais, não é apenas um diferencial de mercado para a organização conseguir vender e lucrar mais, é um pré-requisito para conseguir colocar seus produtos em um Mercado Globalizado.

Na área de software, observamos a necessidade de uma maior preocupação com a qualidade. Mas afinal, o que é qualidade?

Trabalharemos com os seguintes conceitos:
- Qualidade é estar em conformidade com os requisitos dos clientes
- Qualidade é antecipar e satisfazer os desejos dos clientes
- Qualidade é escrever tudo o que se deve fazer e fazer tudo o que foi escrito

A norma brasileira NBR-ISO-8402, conceitua qualidade como:

*A totalidade das características de uma **entidade***
que lhe confere a capacidade de satisfazer
*às **necessidades explícitas e implícitas***

Podemos notar que esta conceituação formal nos remete a alguns conceitos acessórios: **entidades**, **necessidades explícitas** e **implícitas**.

- A **entidade** é o produto do qual estamos falando, que pode ser um bem ou um serviço.
- As **necessidades explícitas** são as próprias condições e objetivos propostos pelo produtor.
- As **necessidades implícitas** incluem as diferenças entre os usuários, a evolução no tempo, as implicações éticas, as questões de segurança e outras visões subjetivas.

Um exemplo simples, a qualidade de um prato de comida (a entidade, o produto) está relacionada com a satisfação de necessidades (requisitos) tais como: sabor, aparência, temperatura, rapidez no serviço, preço, higiene, valor nutricional, etc. Para avaliar a qualidade de um produto, deve-se fazer uma lista destas necessidades e analisar cada uma delas.

1.2 – ABORDAGENS

O tema qualidade quando associado à produtividade geralmente implica em abordagens diferentes, tais como: tecnologia competente, economia atraente, mudança no papel das pessoas e das empresas e capacitação pessoal.

Como *tecnologia competente* focada em informática, entende-se a utilização de modernas ferramentas, tais como sistemas operacionais, linguagens de programação e respectivos recursos atualizados, hardware, software, telecomunicações e gestão de dados seguros e confiáveis, etc. Embora as organizações devam fazer uso dos recursos disponíveis na busca pela qualidade e produtividade, a utilização dos instrumentos citada favorecerá tal busca. É recomendável, no entanto, que se conheça o estado da arte dos instrumentos a serem utilizados, que se avalie corretamente as necessidades da empresa e que seja respeitada sua realidade econômica e financeira através de uma criteriosa avaliação dos custos e benefícios que poderão ser obtidos com as tecnologias disponíveis.

Por *economia atraente*, entende-se os investimentos mínimos adequados à realidade da empresa, respeitando a situação atual e sua capacidade econômica e financeira.

A *mudança no papel* das pessoas, da empresa e da área de Tecnologia de Informação pode ser compreendida como uma Unidade Departamental aberta, transparente, utilizadora de modernas e estruturadas metodologias para a realização do trabalho.

Por *capacitação profissional*, entende-se uma formação atualizada e completa dos profissionais que atuam na empresa, ou seja, a exigência de uma excelente formação técnica, do negócio em questão e do comportamento humano.

Na área de informática é interessante ressaltar que se tem tido muita produção, muita eficiência, mas a custa de baixa produtividade e, muitas vezes, baixa qualidade.

1.3 – MAIS ALGUNS CONCEITOS

a) Qualidade

- conformidade com os requisitos, ou adaptabilidade ao uso, adequação ao cliente/ usuário;
- atendimento perfeito de forma confiável (sem defeitos), acessível (baixo custo), de forma segura e tempestiva, às necessidades do cliente;
- é a ausência de desperdício.

Obs.: a qualidade de um software ou sistema de informação pode ser observada pela adequação à empresa, ao usuário/cliente e ao atendimento a padrões pré-definidos para tal qualidade.

b) Produtividade

- é a relação entre recursos disponíveis consumidos e os resultados obtidos;
- é produzir cada vez mais e melhor, com máxima satisfação das necessidades dos clientes;
- é diferente de produção, que é simplesmente a quantidade produzida, sem valor de uso.

c) Competitividade

- ter maior produtividade que os concorrentes, garantindo a sobrevivência da empresa, perenidade, continuidade ao longo do tempo e satisfação dos clientes internos e externos.

Obs.: A Qualidade, a produtividade e a competitividade estão interligadas e são interdependentes pois a competitividade decorre da produtividade, e esta se baseia na qualidade.

d) Inteligências Competitiva e Empresarial

- Inteligência Competitiva é um processo sistematizado de obtenção de informações externas para garantir a Inteligência Empresarial;
- Inteligência Empresarial é um sistema de monitoramento do ambiente interno e externo da empresa para garantir a efetividade, qualidade, produtividade, modernidade, perenidade e rentabilidade da organização.

e) Qualidade em Software ou em Sistemas de Informação

- conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software e/ou sistema que seja desenvolvido de forma profissional.
- atendimento aos padrões ISO e dos usuários/clientes.

Os preceitos para um Software de Qualidade, podem ser:

- defeito zero → principalmente importante para aplicações ou sistemas críticos;
- requisitos funcionais adequados → as funções e os requisitos devem ser adequados à satisfação do usuário/cliente;
- codificação estruturada e legível → para que qualquer outro profissional possa efetuar manutenções sem dificuldades;
- desempenho satisfatório → os tempos de resposta são adequados;
- custo adequado → onde o investimento seja justo em função dos resultados e retornos esperados;
- desenvolvimento rápido e produtivo → onde a equipe de desenvolvimento e os usuários/clientes elaboram um planejamento adequado e satisfatório;
- facilidades para os usuários/clientes → quando for o caso, para que eles possam manipular o software ou o sistema de forma acessível.

f) Eficiência

- significa fazer certo as coisas e com desempenho;
- está baseada em métodos, meios e tempos.

g) Eficácia

- significa fazer as coisas certas;
- está baseada em resultados e produtos.

h) Efetividade

- significa fazer a coisa certa e da maneira correta;
- é a soma da eficiência e da eficácia;
- está baseada na regularidade, praticidade, durabilidade e constância.

i) Economicidade

- significa a ausência de desperdício de recursos;
- é a efetividade com custo adequado à realidade empresarial;
- está baseada na relação *custo* X *benefício.*

1.4 - CERTIFICAÇÃO DE QUALIDADE

Um aspecto interessante da qualidade é que não basta que ela exista. Ela deve ser **reconhecida** pelo cliente. Por causa disso, é necessário que exista algum tipo de **certificação** oficial, emitida com base em um **padrão**. Você provavelmente já conhece alguns certificados mais comuns:

- O selo do SIF de inspeção da carne;
- O selo da ABIC nos pacotes de café;
- A classificação em estrelas dos hotéis;e
- Os certificados de qualidade da série ISO-9000.

Inúmeras de empresas, em suas campanhas de marketing, falam de sua **certificação ISO-9000**. Isto nada mais é do que um padrão de qualidade (reconhecido mundialmente) pelo qual esta empresa foi avaliada e julgada.

Para que seja possível realizar uma **avaliação** e um **julgamento**, é necessário haver um padrão ou norma. Existem alguns organismos normalizadores que são reconhecidos mundialmente:
. ISO - International Standardization Organization
. IEEE - Instituto de Engenharia Elétrica e Eletrônica
. ABNT - Associação Brasileira de Normas Técnicas

A norma ISO-9000, por exemplo, foi criada pela ISO para permitir que todas as empresas do mundo possam avaliar e julgar sua qualidade. Existindo um **padrão único mundial**, uma empresa do Brasil, mesmo não tendo nenhum contato com uma outra empresa na Europa, pode garantir a ela a qualidade de seu trabalho.

A **Certificação** é a emissão de um documento oficial indicando a conformidade com determinada norma ou padrão. Obviamente, antes da emissão do certificado, é preciso realizar todo um processo de avaliação e julgamento de acordo com uma determinada norma. Embora uma empresa possa ser avaliada por seus próprios clientes, o termo **Certificação** costuma ser aplicado apenas quando efetuado por uma empresa independente e idônea, normalmente especializada neste tipo de trabalho.

No Brasil, o INMETRO é o órgão do governo responsável pelo credenciamento destas instituições que realizam a certificação de sistemas de qualidade.

1.5 - PORQUÊ SE PREOCUPAR COM A QUALIDADE DE SOFTWARE?

As organizações buscam destacar-se no competitivo mercado mundial de software através de produtos e serviços de boa qualidade.

As seguintes razões devem ser levadas em conta:

- Qualidade é competitividade: a única maneira de diferenciar o produto do competidor é pela qualidade do software e do suporte que é fornecido juntamente. Como o mercado amadurece, usuários não querem apenas que a empresa fale que tem qualidade, mas que mostre a todos a sua qualidade através de Certificação internacional. Não ter certificação pode acarretar desvantagem competitiva.
- Qualidade é essencial para a sobrevivência: Clientes estão exigindo qualidade. Se a empresa não tiver habilidade de sobreviver em um mercado altamente competitivo, ela está em débito com o mercado. A maioria das grandes organizações está reduzindo o número de fornecedores, e um meio de escolher os fornecedores é verificando quais deles têm certificações de qualidade.
- Qualidade é essencial para o mercado internacional: O mercado de software está cada vez mais global. A habilidade das empresas de mostrarem qualidade, eventualmente as colocam no mercado global. O mercado local é vulnerável a produtos importados que, normalmente, têm mais qualidade.
- Qualidade é custo/benefício: um sistema de qualidade direciona para o aumento da produtividade e permanente redução de custos, habilitando o gerenciamento para reduzir a correção de defeitos dando ênfase à prevenção. Todas as empresas sabem que corrigir defeitos após o desenvolvimento do software é mais dispendioso do que corrigi-los antes. Prevenir defeitos primeiramente pode resolver muita coisa depois e economizar bastante.
- Qualidade retém consumidores e aumenta lucros: pouca qualidade normalmente custa muito mais do que contratar mais desenvolvedores e ainda continuar sem qualidade. A maioria dos consumidores não tolerará falta de qualidade e irão procurar

outros desenvolvedores. Mais qualidade aumenta a satisfação dos consumidores e assegura os que já são clientes há mais tempo.

1.4.1 - Qualidade e o desenvolvimento de software

Desenvolvimento de software é um processo no qual os desenvolvedores buscam traduzir os pré-requisitos do cliente em software. A qualidade do produto final está diretamente ligada à qualidade dos processos utilizados para o desenvolvimento.

A comunidade de desenvolvedores de software e os clientes exigem que o desenvolvimento do produto deve ser feito de maneira muito rápida. O ciclo de vida do produto, no caso o software, é agora um negócio crítico para muitos desenvolvedores.

Os consumidores de software necessitam de produtos cada vez melhores e mais rápidos de serem desenvolvidos para aumentar a sua competitividade no mercado global.

Se estes objetivos forem cumpridos, o desenvolvimento de software, deve:

- Utilizar as melhores práticas da engenharia de software;
- Ser operado por pessoal treinado com responsabilidades e instruções;
- Dar ênfase à prevenção de defeitos assim que forem detectados;
- Gerar registro para demonstrar efetividade; e
- Utilizar destes registros para aumentar a performance no futuro.

1.4.2 - O que é um Sistema de Qualidade?

O termo "sistema de qualidade" é utilizado internacionalmente para descrever um processo que garante e demonstra a qualidade dos produtos e serviços ofertados pela empresa.

A padronização ISO 9000 define e descreve o que é requerido ou satisfatório em um sistema de qualidade contendo componentes de desenvolvimento. As padronizações existentes para garantir a qualidade de software serão estudadas mais adiante.

Além das padronizações ISO, muitas outras organizações internacionais e nacionais promovem padrões que descrevem sistemas de qualidade para serem aplicados em sistemas de desenvolvimento e suporte, como por exemplo o CMM (Capability Maturity Model).

Um sistema de qualidade dá grande ênfase à correção de erros. É muito útil corrigir os erros durante o início do ciclo de vida do sistema. Melhor ainda, seria anular erros antes mesmo de acontecerem.

Um sistema de qualidade de sucesso inclui maneiras de registrar os erros para determinar as causas e agir de acordo com o erro eliminando suas causas.

Em resumo, um sistema de qualidade é tudo que o gerenciamento utiliza para garantir e demonstrar a qualidade do software e do serviço de suporte. O sistema de qualidade é o trabalho completo, incluindo política, procedimentos, ferramentas e recursos, incluindo humanos e tecnológicos.

1.4.3 - Qualidade do Produto x Qualidade do Processo

Uma das evoluções mais importantes, constatadas no estudo da qualidade está em notar que a **qualidade do produto** é algo bom, mas que **qualidade do processo** de produção é ainda mais importante.

No caso do prato de comida, citado anteriormente, podemos dizer mais sobre a qualidade observando como o prato foi preparado do que analisando o produto final; não conseguiremos ter certeza da higiene ou do valor nutricional apenas comendo o prato.

Hoje em dia, podemos consultar normas e padrões tanto para produtos quanto para processos. Obviamente, os certificados mais valiosos são aqueles que certificam o processo de produção de um produto e não aqueles que simplesmente certificam o produto. Entretanto, é comum encontrar empresas que perseguem os dois tipos de certificação de qualidade.

Unidade 2 – Qualidade de Software

2.1 - PROPOSIÇÕES DA QUALIDADE DE SOFTWARE:

1) Qualidade é uma forma de sucesso para o mercado de software, como para qualquer outro;
2) A maneira mais barata de aumentar a produtividade é aumentar a qualidade do software;
3) A qualidade do suporte ao produto é tão importante quanto à qualidade do próprio software; o ambiente de suporte deve ter tanta engenharia quanto o ambiente de desenvolvimento;
4) Para alcançar a qualidade de software, as pessoas e a cultura são tão importantes quanto à tecnologia;
5) O único caminho seguro para aumentar a qualidade do software é melhorar os processos (inclui pessoal, facilidades, equipamentos, tecnologia e metodologia);
6) Aumento de processos é normalmente desnecessário e requer que o gerente demonstre compromisso e liderança;
7) Qualidade e melhoramento dos processos são de difíceis esforços. Entretanto, é sempre possível realizar algo um pouco melhor, um pouco mais rápido e um pouco mais barato;
8) Um sistema de qualidade compatível com ISO9000 é um bom alvo para muitas organizações, mas não para todas;
9) Um sistema de qualidade para uma organização deve ser medido de acordo com suas necessidades e circunstâncias ou não será eficiente

Um sistema de qualidade de software eficiente que se utilize das boas práticas da engenharia de software, deve se basear em princípios de qualidade, de gerência e de engenharia:

Princípios de qualidade

- Tentar prevenir defeitos ao invés de consertá-los;
- Ter certeza de que os defeitos que forem encontrados, serão corrigidos o mais rápido possível;
- Estabelecer e eliminar as causas, bem como os sintomas dos defeitos;
- Auditar o trabalho de acordo com padrões e procedimentos previamente estabelecidos;

Princípios de gerência

- Definir regras e responsabilidades;
- Planejar o trabalho;
- Trilhar o progresso através de planos e corrigir quando necessário;
- Refinar o plano sempre e progressivamente;

Princípios de engenharia

- Analisar o problema antes de desenvolver a solução;
- Quebrar problemas complexos em problemas menores;
- Garantir que os problemas menores ou subproblemas estejam unidos pelo controle de seus relacionamentos;

Agora que temos algum conhecimento sobre qualidade e sobre como ela poderia ser avaliada, vamos tentar aplicar estes conceitos aos produtos de software e ao processo de desenvolvimento de software.

Inicialmente, vamos encontrar um grande problema: muitas pessoas acham que criar programas é uma arte que não pode seguir regras, normas ou padrões. Isto acontece principalmente porque:

- Produtos de software são bastante complexos; geralmente, mais do que o hardware onde executam;
- Software não tem produção em série. Seu custo está no projeto e desenvolvimento;
- Software não se desgasta e nem se modifica com o uso;
- A representação de software em gráficos e diagramas não é precisa;
- A Engenharia de Software ainda é uma tecnologia em evolução;
- Há divergências, entre os profissionais da área, sobre o que é Qualidade de Software

Apesar de tudo isso, é preciso entender que o problema não está no Software em si, mas na forma como as pessoas têm desenvolvido software até os dias de hoje.

Alguns desenvolvedores de software continuam a acreditar que a qualidade de software é algo que só começa a preocupar depois que o código foi gerado. A Garantia da Qualidade de Software (S*oftware Quality Assurance*, SQA) deve ser considerada como uma atividade a ser aplicada ao longo de todo o processo de desenvolvimento de software. Neste ponto, é necessário conhecer mais dois conceitos importantes:

- *Qualidade de projeto:* refere-se a características que os projetistas especificam para um certo produto. No desenvolvimento de software, abrange os requisitos, as especificações e o projeto do sistema.
- *Qualidade de Conformidade:* é o grau com que as especificações de projeto são seguidas durante a "fabricação". No desenvolvimento de software, é um assunto concernente, principalmente, à implementação.

2.2 – NORMAS

Muitas instituições se preocupam em criar Normas para permitir a correta avaliação de Qualidade tanto de produtos de Software quanto de processos de desenvolvimento de Software. Apenas para ter uma visão geral, observe o quadro abaixo com as principais Normais nacionais e internacionais nesta área:

Norma	Comentário
ISO 9126	Características da Qualidade de produtos de Software.
NBR 13596	Versão brasileira da ISO 9126
ISO 14598	Guias para a avaliação de produtos de Software, baseados na utilização prática da Norma ISO 9126
ISO 12119	Características de Qualidade de pacotes de Software (Software de prateleira, vendido com um produto embalado)

IEEE P1061	Standard for Software *Quality Metrics Methodology* (produto de Software)
ISO 12207	Software *Life Cycle Process*. Norma para a Qualidade do processo de desenvolvimento de Software.
NBR ISO 9001	Sistemas de Qualidade – Modelo para garantia de Qualidade em Projeto, Desenvolvimento, Instalação e Assistência Técnica (processo)
NBR ISO 9000-3	Gestão de Qualidade e garantia de Qualidade. Aplicação da Norma ISO 9000 para o processo de desenvolvimento de Software.
NBR ISO 10011	Auditoria de Sistemas de Qualidade (processo)
CMM	*Capability Maturity Model*. Modelo da SEI (Instituto de Engenharia de Software do Departamento de Defesa dos EEUU) para avaliação da Qualidade do processo de desenvolvimento de Software. Não é uma Norma ISO, mas é muito bem aceita no mercado.
SPICE ISO 15504	Projeto da ISO/IEC para avaliação de processo de desenvolvimento de Software. Ainda não é uma Norma oficial ISO, mas o processo está em andamento.

2.3 - QUALIDADE DE PRODUTOS DE SOFTWARE - ISO 9126

Quando se pensa em qualidade de um "produto físico", é fácil imaginar padrões de comparação, provavelmente ligado às dimensões do produto ou alguma outra característica física. Quando se trata de software, é difícil definir exatamente o que é a qualidade.

A **ISO** publicou, em 1991, uma norma que representa a atual padronização mundial para a qualidade de produtos de software - ISO/IEC 9126. Ela é uma das mais antigas da área de qualidade de software e sua tradução para o Brasil, foi publicada em agosto de 1996 como NBR 13596.

Estas normas listam o conjunto de características que devem ser verificadas em um software para que ele seja considerado um "software de qualidade". São seis grandes grupos de características, cada um dividido em algumas sub-características.

Característica	Sub-característica	Pergunta chave para a sub-característica
Funcionalidade (satisfaz as necessidades?)	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Faz o que foi proposto de forma correta?
	Interoperabilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis, etc.?
	Segurança de acesso	Evita acesso não autorizado aos dados?
Confiabilidade (é imune a falhas?)	Maturidade	Com que freqüência apresenta falha?
	Tolerância a falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falha?
Usabilidade (é fácil de usar?)	Inteligibilidade	É fácil entender o conceito e a aplicação?
	Apreensibilidade	É fácil aprender a usar?
	Operacionalidade	É fácil de operar e controlar?

Eficiência (é rápido e "enxuto"?)	Tempo	Qual é o tempo de resposta, a velocidade de execução?
	Recursos	Quanto recurso usa? Durante quanto tempo?
Manutenibilidade (é fácil de modificar?)	Analisabilidade	É fácil de encontrar uma falha, quando ocorre?
	Modificabilidade	É fácil modificar e adaptar?
	Estabilidade	Há grande risco quando se faz alterações?
	Testabilidade	É fácil testar quando se faz alterações?
Portabilidade (é fácil de usar em outro ambiente?)	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capac. para ser instalado	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capac. Para substituir	É fácil usar para substituir outro?

2.4 – GUIAS PARA AVALIAÇÃO DA QUALIDADE - ISO 14598

As características e sub-características da norma ISO/IEC 9126 apenas começaram o trabalho de avaliar a qualidade de um software. Faltava definir, em detalhes, como atribuir um conceito para cada item. Afinal, sem uma padronização, que valor teria uma avaliação?

A ISO, consciente do problema, utiliza um conjunto de Guias para a Avaliação da Qualidade segundo a norma ISO/IEC 9126. Estes guias descrevem, detalhadamente, todos os passos para que se avalie um software.

Esta trata o processo de avaliação em grande detalhe. Ela leva em conta a existência de três grupos interessados em avaliar um software, o que define os três tipos básicos de certificação:

Certificação	Quem realiza	Finalidade
de 1ª. parte	Empresas que desenvolvem software	Melhorar a qualidade de seu próprio produto
de 2ª. parte	Empresas que adquirem software	Determinar a qualidade do produto que irão adquirir
de 3ª. parte	Empresas que fazem certificação	Emitir documento oficial sobre a qualidade de um software

Esta norma se constitui, na verdade, de seis documentos distintos, relacionados entre si, como demonstrado a seguir:

Norma	Nome	Finalidade
14598-1	Visão Geral	Ensina a utilizar as outras normas do grupo
14598-2	Planejamento e Gerenciamento	Sobre como fazer uma avaliação, de forma geral
14598-3	Guia para Desenvolvedores	Como avaliar sob o ponto do vista de quem desenvolve
14598-4	Guia para Aquisição	Como avaliar sob o ponto de vista de quem vai adquirir

14598-5	Guia para Avaliação	Como avaliar sob o ponto de vista de quem certifica
14598-6	Módulos de Avaliação	Detalhes sobre como avaliar cada característica

Em resumo, esta norma complementa a ISO/IEC 9126 e permite uma avaliação padronizada das características de qualidade de um software.

É importante notar que, ao contrário da 9126, a 14598 vai a detalhes mínimos, incluindo modelos para relatórios de avaliação, técnicas para medição das características, documentos necessários para avaliação e fases da avaliação. Como um exemplo, observe um modelo de relatório de avaliação, segundo um anexo da norma 14598-5:

Seção	Itens
1 - Prefácio	Identificação do avaliador Identificação do relatório de avaliação Identificação do contratante e fornecedor
2 - Requisitos	Descrição geral do domínio de aplicação do produto Descrição geral dos objetivos do produto Lista dos requisitos de qualidade, incluindo - Informações do produto a serem avaliadas - Referências às características de qualidade - Níveis de avaliação
3 - Especificação	Abrangência da avaliação Referência cruzada entre os requisitos de avaliação e os componentes do produto Especificação das medições e dos pontos de verificação Mapeamento entre a especificação das medições com os requisitos de avaliação
4 - Métodos	Métodos e componentes nos quais o método será aplicado
5 - Resultado	Resultados da avaliação propriamente ditos Resultados intermediários e decisões de interpretação Referência às ferramentas utilizadas

2.5 – Qualidade de Pacotes de Software - ISO 12119

Esta norma foi publicada em 1994 e trata da avaliação de pacotes de software, também conhecidos como "software de prateleira". Além de estabelecer os requisitos de qualidade para este tipo de software, ela também destaca a necessidade de instruções para teste deste pacote, considerando estes requisitos. A norma divide-se em itens, da seguinte forma:

Item	Descrição
1. Escopo	
2. Definições	
3. Requisitos de qualidade	
3.1. Descrição do Produto	Descreve o produto, de forma a ajudar o comprador em potencial, servindo como base para testes. Cada declaração deve ser correta e testável. Deve incluir declarações sobre funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

3.2. Documentação do usuário	Deve ser completa, correta, consistente, fácil de entender e capaz de dar uma visão geral do produto.
3.3. Programas e dados	Descreve em detalhes cada uma das funções do software, incluindo declarações sobre funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.
4. Instruções para teste	
4.1. Pré-requisitos de teste	Lista de itens necessários ao teste, incluindo documentos incluídos no pacote, componentes do sistema e material de treinamento.
4.2. Atividades de teste	Instruções detalhadas sobre os procedimentos de teste, inclusive instalação e execução de cada uma das funções descritas.
4.3. Registro de teste	Informações sobre como os testes foram realizados, de tal forma a permitir uma reprodução destes testes. Deve incluir parâmetros utilizados, resultados associados, falhas ocorridas e até a identidade do pessoal envolvido.
4.4. Relatório de teste	Relatório incluindo: identificação do produto, hardware e software utilizado, documentos utilizados, resultados dos testes, lista de não conformidade com os requisitos, lista de não conformidade com as recomendações, datas, etc.

Um dos grandes méritos desta norma está na profundidade com que são descritas cada uma das características e sub-características mencionadas na norma 9126. A norma inclui detalhes que devem estar presentes no produto, tais como:

- Documentação do usuário de fácil compreensão.
- Um sumário e um índice remissivo na documentação do usuário.
- Presença de um Manual de instalação com instruções detalhadas.
- Possibilidade de verificar se uma instalação foi bem sucedida.
- Especificação de valores limites para todos os dados de entrada.
- Operação normal mesmo quando os dados informados estiverem fora dos limites especificados.
- Consistência de vocabulário entre as mensagens e a documentação.
- Função de auxílio (help) com recursos de hipertexto.
- Mensagens de erro com informações necessárias para a solução da situação.
- Diferenciação dos tipos de mensagem: confirmação, consulta, advertência e erro.
- Clareza nos formatos das telas de entrada e relatórios.
- Capacidade de reverter funções de efeito drástico.
- Alertas claros para as conseqüências de uma determinada confirmação.
- Identificação dos arquivos utilizados pelo programa.
- Identificação da função do programa que está sendo executada no momento.
- Capacidade de interromper um processamento demorado.

2.6 – A SÉRIE ISO 9000:

Esta série é um conjunto de normas da ISO que define padrões para garantia e gerenciamento da qualidade. Algumas destas normas estão descritas abaixo:

Norma	Trata de
ISO 9001	Modelo para garantia da qualidade em projeto, desenvolvimento, produção, instalação e assistência técnica.
ISO 9002	Modelo para garantia da qualidade em produção e instalação

ISO 9003	Modelo para garantia da qualidade em inspeção e ensaios finais
ISO 9000-1	Diretrizes para escolher entre as normas ISO 9001, 9002 e 9003
ISO 9000-3	Orientação para a aplicação da ISO 9001 em Software

Entre as normas 9001, 9002 e 9003, a primeira é a que mais se adequa ao desenvolvimento e manutenção de software. Como toda norma deste grupo, ela é usada para garantir que um fornecedor atende aos requisitos especificados nos diversos estágios do desenvolvimento. Estes estágios incluem projeto, desenvolvimento, produção, instalação e suporte.

A norma ISO 9000-3 (não confundir com a ISO 9003) traz os roteiros para aplicar a ISO 9001 especificamente na área de desenvolvimento, fornecimento e manutenção de software. Todas as orientações giram em torno de uma "situação contratual", onde uma outra empresa contrata a empresa em questão para desenvolver um produto de software. Na tabela a seguir estão os processos definidos na ISO 9000-3:

Grupo	Atividade
Estrutura do Sistema de Qualidade	Responsabilidade do fornecedor Responsabilidade do comprador Análise crítica conjunta
Atividades do Ciclo de Vida	Análise crítica do contrato Especificação dos requisitos do comprador Planejamento do desenvolvimento Projeto e implementação Testes e validação Aceitação Cópia, entrega e instalação Manutenção
Atividades de Apoio	Gerenciamento de configuração Controle de documentos Registros da qualidade Medição Regras, convenções Aquisição Produto de software incluído Treinamento

O processo de certificação de uma empresa de software segundo as normas ISO 9001/9000-3 segue um conjunto de passos bem definidos:

- A empresa estabelece o seu sistema de qualidade.
- A empresa faz uma solicitação formal a um órgão certificador, incluindo detalhes do negócio da empresa, escopo da certificação solicitada e cópia do manual de qualidade.
- O órgão certificador faz uma visita à empresa, colhe mais dados e explica o processo de certificação.
- O órgão certificador verifica se a documentação do sistema de qualidade está de acordo com a norma ISO e envia uma equipe à empresa com fins de auditoria. Nesta visita, será verificado se todos na empresa cumprem o que está documentado no manual de qualidade.

- O órgão certificador emite o certificado de qualidade.
- O órgão certificador realiza visitas periódicas à empresa para assegurar que o sistema continua sendo efetivo.

2.7 - ISO 12207 – PROCESSO DE CICLO DE VIDA DE SOFTWARE

Este padrão formaliza a arquitetura do ciclo de vida do software, que é um assunto básico em Engenharia de Software e também em qualquer estudo sobre Qualidade do Processo de Software. Esta norma possui mais de 60 páginas e detalha os diversos processos envolvidos no ciclo de vida do software.

Estes processos estão divididos em três classes: Processos Fundamentais, Processos de Apoio e Processos Organizacionais. Segue a lista completa dos processos na tabela abaixo:

Processos Fundamentais	**Início e execução do desenvolvimento, operação ou manutenção do software durante o seu ciclo de vida.**
Aquisição	Atividades de quem um software. Inclui: definição da necessidade de adquirir um software (produto ou serviço), pedido de proposta, seleção de fornecedor, gerência da aquisição e aceitação do software.
Fornecimento	Atividades do fornecedor de software. Inclui preparar uma proposta, assinatura de contrato, determinação recursos necessários, planos de projeto e entrega do software.
Desenvolvimento	Atividades do desenvolvedor de software. Inclui: análise de requisitos, projeto, codificação, integração, testes, instalação e aceitação do software.
Operação	Atividades do operador do software. Inclui: operação do software e suporte operacional aos usuários.
Manutenção	Atividades de quem faz a manutenção do software.
Processos de Apoio	Auxiliam um outro processo.
Documentação	Registro de informações produzidas por um processo ou atividade. Inclui planejamento, projeto, desenvolvimento, produção, edição, distribuição e manutenção dos documentos necessários a gerentes, engenheiros e usuários do software.
Gerência de Configuração	Identificação e controle dos itens do software. Inclui: controle de armazenamento, liberações, manipulação, distribuição e modificação de cada um dos itens que compõem o software.
Garantia da Qualidade	Garante que os processos e produtos de software estejam em conformidade com os requisitos e os planos estabelecidos.
Verificação	Determina se os produtos de software de uma atividade atendem completamente aos requisitos ou condições impostas a eles.
Validação	Determina se os requisitos e o produto final (sistema ou software) atendem ao uso específico proposto.
Revisão Conjunta	Define as atividades para avaliar a situação e produtos de uma atividade de um projeto, se apropriado.

Auditoria	Determina adequação aos requisitos, planos e contrato, quando apropriado.
Resolução de Problemas	Analisar e resolução dos problemas de Qualquer natureza ou fonte, descobertos durante a execução do desenvolvimento, operação, manutenção ou outros processos. .
Processos Organizacionais	Implementam uma estrutura constituída de processos de ciclo de vida e pessoal associado, melhorando continuamente a estrutura e os processos.
Gerência	Gerenciamento de processos.
Infra-estrutura	Fornecimento de recursos para outros processos. Inclui: hardware, software, ferramentas, técnicas, padrões de desenvolvimento, operação ou manutenção.
Melhoria	Atividades para estabelecer, avaliar, medir, controlar e melhorar um processo de ciclo de vida de software.
Treinamento	Atividades para prover e manter pessoal treinado.

A norma detalha cada um dos processos acima. Ela define ainda como eles podem ser usados de diferentes maneiras por diferentes organizações (ou parte destas), representando diversos pontos de vista para esta utilização. Cada uma destas visões representa a forma como uma organização emprega estes processos, agrupando-os de acordo com suas necessidades e objetivos.

As Visões têm o objetivo de organizar melhor a estrutura de uma empresa, para definir suas gerências e atividades alocadas às suas equipes. Existem cinco visões diferentes: contrato, gerenciamento, operação, engenharia e apoio.

A ISO/IEC 12207 é a primeira norma internacional que descreve em detalhes os processos, atividades e tarefas que envolvem o fornecimento, desenvolvimento, operação e manutenção de produtos de software. A principal finalidade desta norma é servir de referência para os demais padrões que venham a surgir. Lançada em agosto de 1995, ela é citada em quase todos os trabalhos relacionados à Engenharia de Software desde então, inclusive aqueles relativos à qualidade.

2.8 – Gerenciando uma Campanha de Qualidade

Gerenciar uma campanha de qualidade é mais do que implementar um sistema de qualidade consistindo de um conjunto de técnicas do padrão Iso. É a criação da "cultura da qualidade" que permeia toda a organização.

A seguir, mostraremos algumas idéias práticas que podem vir a ajudar na realização de trabalho com mais qualidade.

Dedicação à satisfação do cliente

Qualidade fala sobre a satisfação do cliente. Produzir um produto de qualidade é uma parte importante para garantir a satisfação e atenção do cliente. Isto inclui:

- Esforçar-se ao máximo para entender as necessidades do cliente no produto e no suporte;
- Escrever um contrato que reflita estas necessidades;
- Prever suporte pós-venda para igualar requisitos ou expectativas;
- Vigiar os contatos com os clientes. A primeira impressão é a que fica.

Dar ênfase a melhoramentos contínuos

O fim da qualidade total nunca será alcançado, será sempre possível fazer as coisas um pouco melhor ou mais rápidas. A melhor coisa a ser feita é a organização se tornar uma "Organização do aprendizado" que constantemente pesquisa e renova recursos e experiências de todo o pessoal envolvido para aumentar a qualidade, reduzir custos e responder rapidamente às necessidades dos clientes. Tendo se tornado uma organização do aprendizado, aprenda como aprender mais rápido do que os concorrentes.

Tratar fornecedores como parceiros de negócios

Esta filosofia de parceiros de negócios pode ser estendida a outros com uma participação no sucesso do seu negócio. As empresas de desenvolvimento devem considerar seus agentes e distribuidores como parceiros do negócio para benefício mútuo. E o mesmo pode ser aplicado a grandes consumidores, que estão procurando relacionamentos.

Decisões em negócios de parcerias não tem razões idealistas, mas tem no gerenciamento do negócio a luz da realidade estratégica e comercial. Eles refletem o fato de o negócio não ser uma coisa qualquer e que é geralmente em seu próprio interesse promover o sucesso dos outros. Em outras palavras é criada uma situação "Vencedor- Vencedor".

Comunicação e time de trabalho

É importante criar uma cultura, na empresa, na qual indivíduos e departamentos acreditem que são seus próprios consumidores dentro da organização. Estes supridores internos tentam entender e suprir as necessidades de seus consumidores internos e estes trabalham com seus supridores internos para ajudá-los.

Responsabilidades e linhas de comunicação devem ser regularmente revistas, e relacionamentos tradicionais entre departamentos devem ser alterados quando for necessário.

Atualizando empregados

O pessoal da empresa não será capaz de produzir um sistema de qualidade e fornecer um suporte de qualidade a menos que eles tenham as habilidades necessárias, conhecimento e recursos. O gerente deve selecionar, em primeiro lugar, o pessoal certo para cada tarefa a ser executada de acordo com suas habilidades e treinamento que eles venham a precisar para conseguir realizar o seu melhor. Isto incentivará o pessoal e gerará um compromisso para um contínuo melhoramento do seu conhecimento e do seu trabalho.

Compromissos da gerência

- Tratar a qualidade como um negócio estratégico;
- Desenvolver uma política para a qualidade;
- Garantir que a política da qualidade será comunicada e implementada através da organização; e
- Regularmente rever a política e suas implementações.

2.9 – IMPLEMENTANDO UM SISTEMA DE QUALIDADE

Como alcançar a qualidade?

Alguns anos atrás, John Ruskin ditou alguns ponteiros quando disse que "***a qualidade nunca é um acidente é sempre o resultado de grandes esforços***". Os dizeres de Ruskin ainda são aplicáveis. Para os negócios modernos, resultados inteligentes devem ser direcionados através de um sistema de qualidade.

2.9.1 – Um Sistema de Qualidade

Um programa de aumento de qualidade conduz ao estabelecimento de um sistema de qualidade que deve ter aspectos técnicos e culturais. É fácil de ver a razão disto: a entidade a ser melhorada consiste em tecnologia e pessoas.

Aspectos Técnicos
Envolvem o desenvolvimento de padrões e técnicas para implementar a qualidade em todas as atividades.

Aspectos culturais
A prática da qualidade deve ser aceita por todas as pessoas da empresa sem exceção. Cada pessoa deve estar consciente da sua responsabilidade pela qualidade. Um programa de educação e treinamento deve ser implementado para conseguir conscientizar a todos na empresa.

2.9.2 – Iniciando um Sistema de Qualidade

Preparar uma política de qualidade

A gerência da empresa deverá começar um projeto de qualidade formulando uma política de qualidade. Esta política deve ser claramente declarada pela "comissão de organização da qualidade" e gerentes que estarão envolvidos no projeto.

A política deverá ser publicada e comunicada para que seja entendida e implementada em todos os setores da empresa.

Estabelecer uma equipe de suporte em qualidade

Uma equipe de suporte à qualidade normalmente compreende um comitê de direção e uma equipe de aumento da qualidade. A gerência sênior da companhia deve estar no comitê de direção.

As tarefas principais do comitê de direção são:

- Definir uma direção estratégica e metas em longo prazo;
- Estabelecer uma equipe de aumento e melhoramento da qualidade e acompanhar sua performance;
- Autorizar e aprovar gastos para o programa de qualidade;
- Prover suporte de alto nível para o programa de qualidade.

O comitê de direção deverá adotar uma equipe de melhoramento da qualidade que será responsável por estabelecer e manter um programa de qualidade em toda a organização.

As atividades específicas são:

- Avaliar as necessidades da organização;
- Desenvolver um sistema de qualidade para a empresa;
- Planejar e monitorar a implementação;

Outras atividades poderão estar sendo desenvolvidas na organização, a saber:

- Comunicar, durante todo o processo, o programa de qualidade ao pessoal da empresa;
- Treinamento e outras atividades para ajudar no suporte ao programa de qualidade;
- Preparar e rever procedimentos e padrões;
- Adotar um ciclo de vida para o desenvolvimento do programa;
- Adotar métodos e ferramentas;
- Estabelecer um programa para medir os processos do software, produtos e serviços.

2.9.3 – Definindo um Programa para a Qualidade

A seguir, mostraremos alguns passos que podem ser utilizados para construir e definir corretamente um programa de implementação da qualidade na empresa.

Avaliar a organização

- Para se avaliar a organização, chama-se um órgão respeitado internacionalmente para que ele avalie os processos da empresa.
- Este órgão, após a verificação, definirá padrões e regras que deverão ser cumpridas para se conseguir o certificado.
- A empresa que estará sendo avaliada, vai reorganizar-se de acordo com as normas apresentadas.
- As normas podem variar de acordo com a necessidade da empresa e serem aplicados padrões internacionais, tais como ISO 9000 ou CMM.

Projetar um sistema de qualidade

Nesta etapa da implementação, a empresa vai definir quais os seus objetivos perante a qualidade que se quer alcançar.

Alguns dos objetivos que as empresas buscam para se adaptar ao mercado, são:

- Reduzir o número de erros e defeitos encontrados na fase de teste e na implantação do software;
- Aumentar a produtividade e reduzir o tempo de desenvolvimento;
- Diminuir o tempo de resposta às requisições do cliente;
- Melhorar as estimativas de custo e agendamento de tarefas para se entregar ao usuário;
- Alcançar a certificação necessária.

Todos os objetivos e partes do projeto devem ser colocados em um manual que, mais tarde, deverá conter todas as resoluções adotadas durante o projeto. Este "Manual da Qualidade" deverá ser publicado e conhecido por todas as pessoas da empresa, para entendimento e retirada de dúvidas. Este manual, poderá conter:

- A política de qualidade e objetivos da empresa;
- A estrutura da organização mostrando as responsabilidades de todos que gerenciam, fazem e verificam o trabalho que afetará a qualidade;
- Uma descrição do ciclo de vida da qualidade na empresa (será visto mais à frente);
- Uma visão geral do sistema de qualidade;
- Relacionamento do sistema de qualidade com o padrão internacional adotado;
- Referência completa a procedimentos e padrões detalhados.

Planejamento e implementação do programa de qualidade

Após a conclusão do "Manual de Qualidade" e sua aprovação pelo pessoal da empresa, uma comissão de organização da qualidade determinará a quantidade de serviço que deverá ser necessário para implementar o programa de qualidade.

A comissão deverá desenvolver um plano de implementação onde serão detalhados tarefas, atividades, marcos e recursos para a implantação. Normalmente, as tarefas mais comuns a serem realizadas são as seguintes:

- implementar um programa cultural;
- Adotar um ciclo de vida;
- Desenvolver um sistema de controle de qualidade;
- Desenvolver e documentar procedimentos e padrões para todas as atividades de cada ciclo da implementação;
- Definir e implementar um programa de medidas;
- Rever e, se necessário, revisar o manual de qualidade;
- Apreciação da qualidade e treinamento do sistema de qualidade;
- Um programa de auditoria da qualidade;
- Revisões gerenciais;
- Avaliação ISO 9000.

2.9.4 – IMPLEMENTANDO UM PROGRAMA CULTURAL

O programa cultural deverá ser implementado em conjunto com o programa técnico. Consiste em conscientizar as pessoas da empresa da necessidade de se ter qualidade em alto nível. Normalmente, se utiliza do treinamento de qualidade e profissional (aprendizado de novas e melhores ferramentas de desenvolvimento) e realização de Workshop para discutir qualidade e necessidades da empresa e das pessoas que estão envolvidas.

Também fazem parte do programa cultural, as iniciativas da gerência e do pessoal da empresa para aumentar o incentivo das pessoas para realizarem uma tarefa melhor e mais bem feita.

As iniciativas da gerência dão ênfase a análise de defeitos e suas causas e a definição de marcos que devem ser alcançados em certo período de tempo.

Já as iniciativas dos funcionários da empresa são voltadas para sugestões de como realizar tarefas e procedimentos de forma a retornar mais qualidade e mais serviço.

2.9.5 – IMPLEMENTANDO O PROGRAMA TÉCNICO

A implementação do programa técnico consiste em:

Adotar um ciclo de vida

Toda empresa de software, grande ou pequena, deve possuir um ciclo de vida para o desenvolvimento do produto. Muitos tipos de ciclo de vida têm sido implementados e testados para atender a determinadas circunstâncias.

A empresa deverá adotar um ciclo de vida e, então adequá-lo às suas necessidades. O ciclo de vida é dividido em etapas. Em cada etapa poderá ser feita uma revisão do que já foi feito e rever o que vai ser feito na próxima etapa.

Desenvolver procedimentos e padrões

Algumas empresas trabalham com o máximo possível de procedimentos e padrões definidos rigorosamente.

É comum aos desenvolvedores não se acostumarem facilmente com estes novos padrões. No caso da empresa estar implantando pela primeira vez um sistema de qualidade, os desenvolvedores estarão sujeitos a não aceitarem o que foi dito e a empresa vai demorar muito tempo tentando conscientizá-las.

Os procedimentos e padrões devem ser identificados da maneira como são utilizados e divulgados pela empresa toda. Estes, normalmente, sofrem a interferência de padrões internacionais como ISO e CMM e devem ser desenvolvidos e utilizados pelas pessoas que já os utilizavam.

Caso a empresa venha implantar novos procedimentos e padrões, eles devem ser feitos de maneira gradual. Se implementar tudo de uma vez, a resistência será maior e a empresa poderá perder por causa disto.

Selecionar métodos e ferramentas.

Os métodos e ferramentas a serem utilizados devem ser padronizados. Esta padronização deverá levar em consideração "dicas" dos desenvolvedores e diretores que estão envolvidos diretamente com o processo de desenvolvimento do software. Estes devem ser utilizados em todos os projetos da empresa e ter uma variação pequena de um projeto para outro.

Aconselha-se que as ferramentas de desenvolvimento devem ser de um mesmo fornecedor para evitar conflitos, atrasos, defeitos inesperados e correções mal feitas para se adaptar as ferramentas diversas ao sistema.

Estabelecer um programa de métricas e medidas de software

É dito que, se você não pode medir, então você não pode gerenciar.

Esta frase pode não estar muito certa, mas se a empresa conseguir "medir" a funcionalidade e a qualidade de seu produto e dos concorrentes, poderá saber se o seu produto está melhor ou pior do que o outro.

Neste caso, as medidas são muito importantes. Quando adotado nas empresas, elas procuram as medidas que devem:

- Refletir os requisitos ou processos dos clientes que direcionam a escolha de pré-requisitos (qualidade não pode ser melhorada otimizando medidas que não relatam as necessidades dos clientes);
- Ser claramente definida. Todos os resultados e medidas devem estar utilizando as mesmas medidas;
- Não utilizar ferramentas caríssimas para coletar dados: às vezes, é melhor coletar dados de uma maneira mais fácil e barata e obter ótimos resultados do que utilizar ferramental caro e obter dados que não são concisos e não obter resultados satisfatórios;

As medidas podem ser utilizadas tanto no projeto quanto nos processos para diferentes propósitos, por exemplo:

- Na fase de projeto, identificar componentes não usuais para um estudo mais detalhado (componentes) que são mais fáceis de darem erros ou defeitos);

- No nível de processos: calibrar uma estimativa do modelo, comparar o uso de diferentes ferramentas e métodos em diferentes projetos, monitorar os avanços progressivos.

Nas empresas, costuma-se, ainda, medir outras atividades relacionadas com o desenvolvimento de software, a exemplificar:

- *Desenvolvimento*
    - Taxas de erros e defeitos;
    - Mudanças nos requisitos e especificações de desenvolvimento;
    - Produtividade;
    - Aderência ao tempo determinado para cada tarefa;
    - Aderência aos gastos previstos.
- *Suporte*
    - Responsabilidade sobre as necessidades do usuário;
    - Freqüência com que o usuário necessita de mudanças.
    - Geral;
    - Custo de baixa qualidade.
- *Treinamento*

Uma grande necessidade da empresa que vai melhorar a qualidade do serviço que ela presta está em treinar o seu pessoal para eles terem consciência do que estão fazendo e conhecer todo o processo da empresa para saberem de suas necessidades e dos outros.

Este treinamento não consiste em treinas as pessoas apenas para receberem novas ferramentas. Elas devem ser treinadas em qualidade, processos novos e antigos, em comunicação, em novas ferramentas e em muitas outras que devem ser especificadas pela empresa de acordo com a necessidade do funcionário e da própria empresa.

2.9.6 – Revisões de Processos e Produtos

As revisões são muito importantes. Elas definem o que foi feito certo, o que foi feito errado e o que precisa ser reparado ou não. É, talvez, a parte do projeto mais importante. É nesta etapa que os desenvolvedores vão descobrir a maioria dos erros e defeitos, que vão ser corrigidos depois. Há, ainda, o teste final que é feito antes de enviar o produto para a implementação no usuário. É importante que não se tenha erros e defeitos quando o software estiver funcionando no cliente, pois ele pode reclamar muito e o projeto todo ficar comprometido.

Revisões do projeto

Todos os projetos, grandes ou pequenos devem ser revistos em sua totalidade para se descobrir o que foi aprendido e o que pode ser melhorado.

Esta revisão pode acontecer uma única vez no final do projeto ou pode acontecer no final de cada fase concluída. Este último pode ser o melhor porque se aprende mais sobre o projeto e já resolve os problemas desta etapa prevendo alguns erros para a próxima etapa.

Revisões da gerência

A gerência tem a máxima responsabilidade para a boa realização do plano de qualidade a ser implantado na empresa. Ela dever rever os processos regularmente, pelo menos uma vez ao ano. Estas revisões devem levantar todas as fontes de informações incluindo:

- Resultado das auditorias, internas ou externas;
- Relatórios das revisões do projeto;

- Relatórios dos serviços ao cliente;
- Mudanças de tecnologia no projeto;
- Avaliações dos clientes sobre a qualidade do serviço;
- Análises de medidas.

Os resultados devem ser consolidados, anexados ao manual e disponibilizados a todos envolvidos, na empresa, para que vejam a evolução do projeto.

2.10 – O FUTURO DA QUALIDADE

Estudiosos da qualidade afirmam que esta terá várias mudanças daqui para frente.

Estas mudanças serão muito grandes porque as empresas, hoje, pensam em qualidade como apenas aquisição de respeito internacional através de certificações e procedimentos considerados corretos e qualitativos.

Mas será que estas empresas apenas conseguem seus certificados e não olham para o produto em si? Elas podem ter certificações mil e continuar desenvolvendo software da mesma maneira que antes e ainda ter várias reclamações de clientes sobre a qualidade do produto em questão.

O que se pensa pelo futuro da qualidade é que as empresas deverão se preocupar em criar um cargo para a gerência da qualidade (CQO[chief quality officer]). Gerentes de hoje, se tornaram gerentes de qualidade amanhã. Estes gerentes não deverão se preocupar em apenas definir regras e procedimentos que devem ser adotados por todos na empresa que trabalham, o que já é feito há muito tempo. Eles deverão concentrar seus esforços nos sistemas de marketing, vendas e suporte porque estes terão mais impacto sobre a satisfação do cliente do que os procedimentos adotados para o desenvolvimento de softwares.

Unidade 3 – Processo e Controle da Qualidade

3.1 - CONCEITOS FUNDAMENTAIS

Controle da qualidade de Software reúne um conjunto de atividades que verificam e garantem uma série de características de qualidade que devem estar presentes no produto para que este atenda às necessidades de seus usuários. Este processo engloba os processos de garantia da qualidade, verificação, validação, revisão conjunta e auditoria da norma ISO 12207 [ISO/IEC 12207].

Todo produto desenvolvido deve ser avaliado segundo o conjunto de características e subcaracterísticas de qualidade definidas na norma ISO 9126 [NBR13596 1996], que estão listadas a seguir:

A) <u>Funcionalidade</u>: conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades especificadas. As funções são as que satisfazem as necessidades explícitas ou implícitas[1].

[1] Este conjunto de atributos caracteriza o que o software faz para satisfazer as necessidades, enquanto que os outros conjuntos caracterizam principalmente quando e como ele faz.

Sub-características:

- *Adequação*: atributos do software que evidenciam a presença de um conjunto de funções e sua apropriação para as tarefas especificadas;
- *Acurácia*: atributos do software que evidenciam a geração de resultados ou efeitos corretos ou conforme acordados;
- *Interoperabilidade*: atributos do software que evidenciam sua capacidade de interagir com sistemas especificados;
- *Segurança de acesso*: atributos do software que evidenciam sua capacidade de evitar o acesso não autorizado, acidental ou deliberado, a programas e dados.
- *Conformidade a Padrões*: atributos do software que fazem com que ele esteja de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à aplicação;

B) Confiabilidade: conjunto de atributos que evidenciam a capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido[2].

Sub-características:

- *Maturidade*: atributos do software que evidenciam a freqüência de falhas por defeitos no software;
- *Tolerância a falhas*: atributos do software que evidenciam sua capacidade em manter o nível de desempenho[3] especificado nos casos de falhas no software ou de violação nas interfaces especificadas[4];
- *Recuperabilidade*: atributos do software que evidenciam sua capacidade de estabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e no tempo e esforço necessários para tal.

C) Usabilidade: conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o software, bem como o julgamento individual desse uso por um conjunto explícito ou implícito de usuários.

Sub-características:

- *Inteligibilidade*: atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade;
- *Apreensibilidade*: atributos do software que evidenciam o esforço do usuário para aprender sua aplicação (por exemplo: controle de operação, entradas, saídas);
- *Operacionalidade*: atributos do software que evidenciam o esforço do usuário para operação e controle de sua operação.

---

[2] Em software não ocorre desgaste ou envelhecimento. As limitações em confiabilidade são decorrentes de defeitos na especificação dos requisitos, projeto ou implementação. As faltas decorrentes desses defeitos dependem de como o produto de software é usado e das opções de programa selecionadas e não do tempo decorrido.

[3] Na especificação do nível de desempenho, deverá estar incluída a capacidade de suportar falhas sem comprometer a segurança.

[4] Um exemplo de violação da interface é a entrada de valores fora das faixas especificadas.

[5] Os recursos podem incluir outros produtos de software, hardware, materiais (por exemplo: papel para impressora. discos flexíveis) e serviços de operação, manutenção ou suporte.

D) <u>Eficiência</u>: conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do software e quantidade de recursos usados, sob condições estabelecidas[5].

<u>Sub-características:</u>
- *Comportamento em relação ao tempo*: atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções;
- *Comportamento em relação aos recursos*: atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas funções.

E) <u>Manutenibilidade</u>: conjunto de atributos que evidenciam o esforço necessário para fazer modificações especificadas no software.

<u>Sub-características:</u>
- *Analisabilidade*: atributos do software que evidenciam o esforço necessário para diagnosticar eficiências ou causas de falhas, ou para identificar partes a serem modificadas;
- *Modificabilidade*: atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais;
- *Estabilidade*: atributos do software que evidenciam o risco de efeitos inesperados, ocasionados por modificações;
- *Testabilidade*: atributos do software que evidenciam o esforço necessário para validar o software modificado.

F) <u>Portabilidade</u>: conjunto de atributos que evidenciam a capacidade do software ser transferido de um ambiente para outro[6].

<u>Sub-características:</u>
- *Adaptabilidade*: atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado;
- *Capacidade para ser instalado*: atributos do software que evidenciam o esforço necessário para sua instalação em um ambiente especificado;
- *Conformidade*: atributos do software que o tornam consoante com padrões ou convenções relacionadas à portabilidade;
- *Capacidade para substituir*: atributos do software que evidenciam sua capacidade e esforço necessário para substituir um outro software, no ambiente estabelecido para este outro software.

Dependendo das especificidades de cada projeto, cada uma destas características poderá ter um peso maior ou menor (ou mesmo não ser pertinente). Poderão, também, ser acrescentadas novas características a este conjunto.

O processo de avaliação a ser adotado é o definido na norma ISO 9126/NBR 13596. A figura 2 mostra os principais passos requeridos para avaliar a qualidade de software. O processo, que deve ser aplicado durante o ciclo de vida, é constituído de três estágios: ***Definição de Requisitos de Qualidade***, ***Preparação da Avaliação*** e ***Procedimento de Avaliação*** [NBR13596 1996]. Esses estágios correspondem a:

[6] O ambiente pode incluir ambiente organizacional, de hardware ou de software.

Definição dos Requisitos de Qualidade

O conjunto de características identificado como necessário para um produto, são os requisitos de qualidade[7] do produto. Para identificação dos requisitos de qualidade de um produto devem ser consideradas as visões de seus diferentes tipos de usuários.

Preparação da Avaliação

O objetivo deste estágio é preparar as bases para a avaliação e envolve:

- *Seleção de métricas:* o modo como as características de qualidade são definidas não permite a sua medição direta. Existe, portanto, a necessidade de se estabelecerem métricas correlacionadas às características do produto de software. Toda particularidade quantificável de software e toda iteração quantificável do software com o seu ambiente que esteja relacionada a uma característica podem ser estabelecidas como uma métrica;
- *Definição dos níveis de pontuação:* o resultado, isto é, o valor medido através das métricas, é mapeado em uma escala. Este valor, entretanto, não mostra o nível de satisfação. Para isso as escalas devem ser divididas em faixas correspondentes aos diversos níveis de satisfação dos requisitos. Uma vez que qualidade se refere a necessidades especificadas, não são possíveis níveis de pontuação genéricos. Eles precisam ser definidos para cada avaliação específica;
- *Definição dos critérios de julgamento:* para se julgar a qualidade de um produto de software, os resultados da avaliação das diversas características devem ser sintetizados. Devem ser usadas tabelas de decisão ou médias ponderadas.

Procedimento de Avaliação

Este estágio é composto de três passos:

- *Medição:* as métricas escolhidas são aplicadas ao produto de software. O resultado é dado em valores nas escalas das métricas;
- *Pontuação:* o nível é determinado para o valor medido, isto é, o resultado da medição é comparado com o nível desejado para o atributo;
- *Julgamento:* o julgamento é o passo final no processo de avaliação, no qual um conjunto de níveis pontuados é sintetizado. O resultado é uma declaração da qualidade do produto de software. A decisão gerencial deve ser tomada com base em critérios gerenciais. O resultado é uma decisão gerencial quanto à aceitação ou rejeição, ou quanto à liberação ou não-liberação do produto de software.

A avaliação da qualidade do produto se fará ao longo de todo o processo de desenvolvimento. As avaliações (intermediárias ou finais) serão realizadas através de reuniões de *walkthrough,* reuniões de inspeção ou testes.

O *Walkthrough* é uma técnica disciplinada onde podem ser avaliados os produtos gerados ao longo do processo de desenvolvimento do software. O processo de realização de um *walkthrough* tem início quando os desenvolvedores informam ao gerente que o produto está concluído. O gerente do projeto verifica se o produto está realmente completo. Caso este o considere completo, é iniciada a avaliação que consiste de duas atividades: *preparação* e *reunião*

[7] Os requisitos expressam as exigências do ambiente pelo produto de software em consideração, e precisam ser definidos antes do desenvolvimento. À medida que um produto de software é decomposto em seus componentes principais, os requisitos derivados do produto como um todo podem ser diferentes para cada componente.

Na *preparação,* o responsável pelo *walkthrough* deve estabelecer a agenda da reunião, gerar cópia do material a ser avaliado e distribuir o material aos avaliadores para que estes preparem a reunião. Os avaliadores preparam a reunião examinando o material e tomando notas sobre os problemas que encontrarem.

A Figura 2, a seguir, mostra o Modelo de Processo de Avaliação [NBR13596 1996]

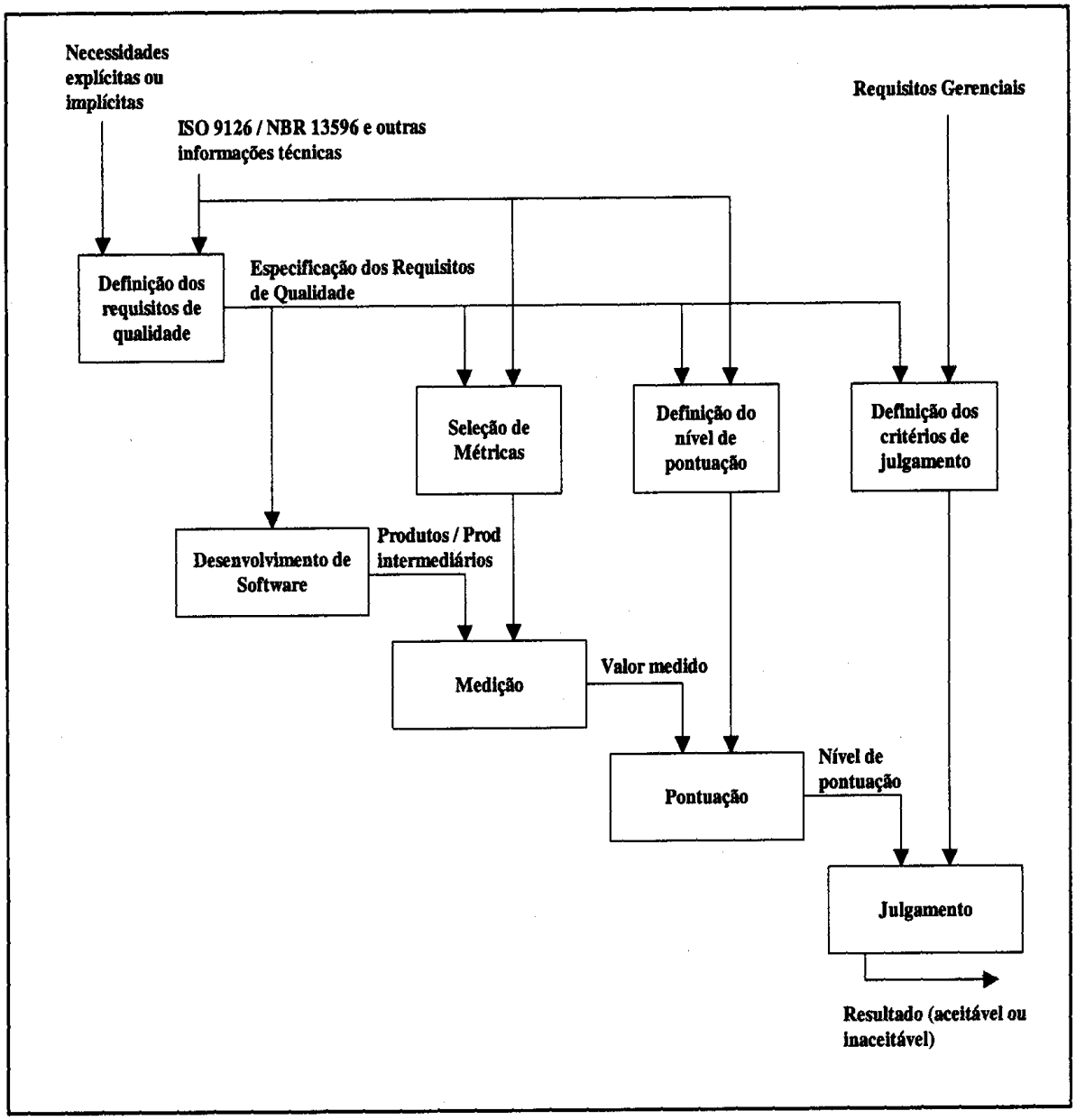

Figura 2 – Modelo de Processo de Avaliação [NBR13596 1996]

A reunião começa com um representante dos desenvolvedores narrando o material a ser avaliado, enquanto os avaliadores fazem observações baseadas na preparação. Durante a reunião, um secretário é nomeado e este deve anotar os erros detectados.

Um moderador coordena a reunião, sendo responsável pelo seu andamento, levantando sugestões e identificando ponderações de consenso. O moderador deverá evitar discussões e divagações, dando ênfase à detecção de anomalias e não à discussão da solução, pois nenhuma correção deve ser feita durante a reunião.

No final da reunião, decide-se sobre a aceitação sem modificações, aceitação com correções, ou rejeição do produto sob avaliação. Se o produto for rejeitado, deverá ser realizada uma nova avaliação.

O secretário deve resumir a reunião em um documento que é denominado Laudo da Reunião de *Walkthrough.*

Uma reunião de inspeção se realiza de forma idêntica ao *walkthrough,* com a diferença de estar baseada em um conjunto de critérios para avaliação, previamente definidos. Os critérios, a serem considerados nas reuniões de inspeção de um projeto, devem estar definidos no Plano de Controle da Qualidade.

As reuniões de *walkthrough* e inspeção devem ser breves (em torno de 2 horas) e nelas devem ser esclarecidas as dúvidas bem como levantadas questões e falhas do produto.

Para tornar eficaz uma reunião de *walkthrough* ou inspeção, deve-se:

- rever o produto e não o desenvolvedor;
- estabelecer uma agenda para a reunião e mantê-la;
- limitar o debate;
- detectar problemas mas não tentar resolvê-los na reunião;
- fazer anotações durante a reunião;
- limitar o número de participantes;
- alocar recursos e espaços no cronograma para a reunião;
- treinar revisores.

Unidade 4 – Perfil Profissional de Qualidade

(extraído de Rezende, Denis A. – Engenharia de Software e Sistemas de Informação – 2ed.- Rio de Janeiro: Brasport, 2002)

O perfil profissional de qualidade, daqueles envolvidos com a tecnologia da informação vai além do domínio técnico.

O mercado de trabalho requer profissionais ecléticos que tenham conhecimentos técnicos, mas que também tenham conhecimentos de negócios e de comportamento humano.

Os aspectos humanos (comportamentais), sociais e políticos da organização estão intimamente ligados com as atividades de sistemas e/ou software.

4.1 – Habilidades Requeridas pelo Mercado

Para atingir a qualidade e produtividade total, requerida pelo mercado de trabalho, há necessidade de uma nova postura no tocante ao perfil profissional, ou seja, o domínio das habilidades técnicas, do negócio e comportamental.

As habilidades do negócio e comportamental são mais difíceis de adquirir, tendo em vista que a formação técnica normalmente é tratada em cursos técnicos, de graduação e pós-graduação. Este perfil será embasado em pesquisas junto a diversas empresas, bem como nas tendências de atuação em informática (SUCESU, 1993).

a. Habilidades técnicas

Adquiridas ao longo da formação técnica do profissional, em cursos acadêmicos e em outros complementares diversos, destacam-se: metodologias, técnicas, ferramentas tecnológicas, instrumentais, incluindo análise de sistemas, programação, lógica, estrutura de dados, algoritmos, arquitetura de computadores, etc.

b. Habilidade de Negócios

Adquiridas ao longo do exercício profissional, desenvolvendo soluções efetivas para as empresas, destacam-se: negócios em questão, funções da administração, funções empresariais, processos, procedimentos, idiomas, etc.

c. Habilidade Comportamentais

Adquiridas ao longo da vida pessoal, na educação, na cultura, filosofia de vida e com os relacionamentos humanos e corporativos, destacam-se: proação (iniciativa, execução e conclusão), criatividade, comunicação, expressão e relacionamento pessoal, espírito de equipe e/ou administração participativa, planejamento pessoal, organização, concentração, atenção, disponibilidade, responsabilidade, etc.

Podem ser conhecidas com mais detalhes nas literaturas de Psicologia Aplicada, Ética, Dinâmica de Grupo, Recursos Humanos contemplando as questões emocionais dos indivíduos. O desafio para conquistar estas habilidades é cotidiano e, principalmente, individual. Os profissionais não podem ser inertes, devem ser ativos e superar a média, que é a ação comum das pessoas sem destaque.

4.2 – TENDÊNCIAS E FORMAS DE ATUAÇÃO

A informática vista como uma ferramenta de trabalho, moderna, transparente e efetiva, deve estar disseminada em toda a empresa, como uma função empresarial.

As empresas estão realizando trabalhos por tarefa ou objetivo, reduzindo os níveis hierárquicos, diminuindo o número de funcionários e exigindo melhor capacitação dos profissionais que nela atuam.

As atuações de informática e de seus profissionais têm as seguintes tendências:

a. Especialista segmentado

- atua nas empresas prestadoras de serviços, em uma determinada atividade, como por exemplo multimídia, redes, treinamento, manutenção de hardware, computação gráfica, marketing, etc.

b. Analista de negócios e/ou informações

- atua nas empresas, em geral, utilizando e integrando os conhecimentos dos recursos de informática em um negócio, como por exemplo, Analista de Sistemas de Recursos Humanos, Comercial, de Produção, de ISO 9000, etc.

c. Engenheiro de software

- profissional de soluções integradas, que utilizam recursos de informática e tecnologias de informação.

d. CIO (Chief Information Officer)

- atua junto a alta administração das empresas, gerindo e fornecendo informações para a tomada de decisões. São, na maioria, ex-gestores da Unidade de Informática.

e. ITM (Information Technology Manager)

- executivo da alta administração que participa das decisões das empresas, ambasado num amplo conhecimento de Tecnologias de Informática e de Informações.

f. Outras tendências

- estão aparecendo em algumas empresas americanas outros perfis, atividades e formas de atuação.

O ***Chief E-Commerce Officer (E-CEO)*** que está vinculado à alta administração da empresa e tem o mesmo status do diretor de informática, que desenvolve algumas atividades especiais. Deve dominar a tecnologia de Internet e as ferramentas de *ecommerce*, mas sua função é mais estratégica, tal como desenvolver soluções para transformar a Internet em diferencial competitivo para empresa, seja por meio da venda direta de produtos ou para interliga-la a fornecedores e distribuidores de forma mais fácil.

O ***Chief Resource Officer (CRO)*** é um gestor executivo de recursos, que surgiu na estrutura das grandes empresas para gerenciar o processo de terceirização dos sistemas. Atua desde a escolha dos prestadores de serviço até o gerenciamento das relações e riscos destas parcerias. Deve ter habilidade de gestionar negócios em diferentes áreas, facilidade para negociar custos e conforto para lidar com mudanças contínuas.

O Chief Knowledge Officer (CKO) - é um gestor executivo de conhecimento de grandes empresas e consultorias, capacitado para administrar o capital intelectual das empresas. Seu perfil é complexo, pois deve ter que provar o quanto é importante armazenar bem e usar as informações para tomada de decisões de toda a empresa. Precisa fazer com que os executivos, além de aprender sobre informações, transformem as mesmas em lucro empresarial.

Essas abreviaturas e/ou denominações estão sempre sendo lançadas no mercado para acompanhar as tendências das atividades da Tecnologia da Informação alinhada aos negócios empresariais.

A forma de atuação de todas essas atividades pode ser como funcionário, prestador de serviços ou consultor, onde o mercado requer destes profissionais que nele atuam muita capacitação, dedicação, qualidade e produtividade nas suas atividades, principalmente porque as empresas, de um modo geral, estão reduzindo seus níveis hierárquicos e o número do quadro de funcionários diretos, muitas delas optando por trabalho pré-definido com objetivos específicos.

4.3 – FATORES HUMANOS E ENGENHARIA HUMANA

Um software ou sistema baseado em computador quase sempre tem um elemento humano, pois uma pessoa pode interagir com o hardware e software, realizando um diálogo que direciona a função do sistema. Anteriormente, a comunicação era mais difícil; hoje, a expressão *user-friendly* (amigável ao cliente/usuário) assumiu um novo significado.

A engenharia de software deve compreender os fatores humanos porque o software é elaborado e usado por pessoas. Nem sempre o conhecimento e os desejos das pessoas são explícitos em requisitos funcionais de software. Dessa forma, o trabalho em equipe é fundamental.

A *engenharia humana* é conhecida como uma etapa muito importante no desenvolvimento de sistemas, pois é uma atividade multidisciplinar que aplica conhecimentos derivados da psicologia e da tecnologia para especificar e projetar uma Interação Homem-Computador (HCI) de alta qualidade.

Entre os muitos fatores que compreendem o elemento humano, pode-se contar: representação da memória e conhecimentos humanos; pensamento e raciocínio; percepção visual; e construção do diálogo humano.

Para facilitar o desenvolvimento de sistemas ou o Projeto de Interfaces, os clientes e/ou usuários podem ser classificados como: participantes (júnior); instruídos e intermitentes (pleno); instruídos e freqüentes (sênior). Estes últimos podem (e devem) contribuir significativamente com as atividades da Engenharia de Software e Sistemas de Informação.

Os aspectos comportamentais e políticos no desenvolvimento de sistemas estão presentes em todas as organizações. A empresa, além de toda a sua estrutura formal e material, também é uma organização social e humana, onde as soluções e decisões não são somente técnicas e racionais.

Sendo assim, inevitavelmente terá conteúdos psicológicos, sociais e políticos, e aí se configurarão relações humanas, de caráter constante, determinadas também pela estruturação de procedimentos e tarefas informais. O sistema social tem restrições externas (família, cultura, comunidade, etc.) que influenciam no próprio sistema interno, do qual emergem os comportamentos de pessoas, grupos e do indivíduo com seu respectivo repertório.

O repertório individual é o conjunto de valores que o indivíduo adquire ao longo da vida, o somatório de suas experiências, valores, credos, costumes, culturas, etc. Este repertório é levado para dentro da empresa, influenciando no trabalho, nas atividades cotidianas e também no desenvolvimento das atividades de Engenharia de Software e Sistemas de Informação.

O sistema informal tem sua origem na necessidade da pessoa humana de conviver com os demais seres humanos. A estrutura emocional, as necessidades, os desejos e a tensão, peculiares a cada pessoa diante de determinada situação de trabalho, também podem refletir em comportamentos muito variados.

Através do processo de afinidade e interação formam-se grupos sem regras formais, mas estabelecendo padrões de prática e comportamento: pares, tríades (três pessoas sendo uma delas um líder informal que mantém o equilíbrio), primários (mais de três pessoas com contato direto e contínuo); e de resistência, com interesses voltados contra os demais grupos e aspectos da organização.

As políticas e o esquema de 'poder' fazem parte da relação entre as pessoas ou grupos, e as principais fontes do poder são: competência – inquestionável sobre determinado tema ou problema; informação – capacidade de influenciar uma direção e/ou um objetivo; apoio – de algo ou alguém; recursos – de quem tem a posse dos recursos disponíveis; autoridade formal – posição hierárquica; liderança pessoal – reconhecida e aceita pelo grupo.

O Conceito e a aplicabilidade do trabalho em conjunto e o princípio de que não existem receitas prontas, sensibilizam o informático a perceber:

- sistema e informática são mais que um conjunto de soluções estritamente técnicas;
- sua ótica profissional não é a única verdadeira;
- o valor da 'alta especialização técnica' pode não significar nada perante um grupo;
- a mudança em sistemas implica em percepções, sentimentos, ações grupais, questões políticas;
- existe um conjunto de valores e uma cultura própria em cada empresa que deve ser considerada;
- a mudança em um sistema também é um fato político e significa realinhamento de forças políticas.

No desenvolvimento de um software, o perfil profissional dos envolvidos, sejam técnicos ou clientes e/ou usuários, contempla pelo menos essas três variáveis ou níveis: de envolvimento, do conhecimento do negócio e da motivação.

Unidade 5 – Capability Maturity Model (CMM)

Este "Modelo de Maturidade da Capacidade" é uma iniciativa do SEI (Software Engineering Institute) para avaliar e melhorar a capacitação de empresas que produzem software.

O projeto CMM foi apoiado pelo Departamento de Defesa do Governo dos Estados Unidos, que, sendo um grande consumidor de software, precisava de um modelo formal que permitisse selecionar os seus fornecedores de software de forma adequada.

Embora não seja uma norma emitida por uma instituição internacional (como a ISO ou o IEEE), esta norma tem tido uma grande aceitação mundial, até mesmo fora do mercado americano. O modelo, publicado em 1992, não é extenso e pode ser obtido na própria Internet com facilidade. O CMM também é chamado de SW-CMM (Software CMM).

5.1 - MATURIDADE

O CMM é um modelo para medição da maturidade de uma organização no que diz respeito ao processo de desenvolvimento de software. A definição do que é "Maturidade" pode ser melhor compreendida através da análise do quadro abaixo:

Organizações maduras	Organizações imaturas
Papéis e responsabilidades bem definidos	Processo improvisado
Existe base histórica	Não existe base histórica
É possível julgar a qualidade do produto	Não há maneira objetiva de julgar a qualidade do produto
A qualidade dos produtos e processos é monitorada	Qualidade e funcionalidade do produto sacrificadas
O processo pode ser atualizado	Não há rigor no processo a ser seguido
Existe comunicação entre o gerente e seu grupo	Resolução de crises imediatas

5.2 - NÍVEIS

O CMM classifica as organizações em cinco níveis distintos, cada um com suas características próprias.

No nível 1, o das organizações mais imaturas, não há nenhuma metodologia implementada e tudo ocorre de forma desorganizada.

No nível 5, o das organizações mais maduras, cada detalhe do processo de desenvolvimento está definido, quantificado e acompanhado e a organização consegue até absorver mudanças no processo sem prejudicar o desenvolvimento.

Observe a seguinte tabela:

Nível CMM	Descrição
1) Inicial	O processo de desenvolvimento é desorganizado e até caótico. Poucos processos são definidos e o sucesso depende de esforços individuais e heróicos.
2) Repetível	Os processos básicos de gerenciamento de projeto estão estabelecidos e permitem acompanhar custo, cronograma e funcionalidade. É possível repetir o sucesso de um processo utilizado anteriormente em outros projetos similares.
3) Definido	Tanto as atividades de gerenciamento quanto de engenharia do processo de desenvolvimento de software estão documentadas, padronizadas e integradas em um padrão de desenvolvimento da organização. Todos os projetos utilizam uma versão aprovada e adaptada do processo padrão de desenvolvimento de software da organização.
4)Gerenciado	São coletadas medidas detalhadas da qualidade do produto e processo de desenvolvimento de software. Tanto o produto quanto o processo de desenvolvimento de software são entendidos e controlados quantitativamente.
5) Otimizado	O melhoramento contínuo do processo é conseguido através de um "feedback" quantitativo dos processos e pelo uso pioneiro de idéias e tecnologias inovadoras.

Uma empresa no nível 1 não dá garantia de prazo, custo ou funcionalidade. No nível 2, a empresa já consegue produzir bons softwares, no prazo e a um custo previsível. O nível 3 garante um excelente nível de qualidade, tanto no produto quanto no processo de desenvolvimento como um todo. Não há, no mundo, muitas empresas que tenham chegado aos níveis 4 e 5...

5.3 - ÁREAS-CHAVE DE PROCESSO (KEY PROCESS AREAS OU KPAS)

Exceto no nível 1, todos os níveis são detalhados em áreas-chave de processo. Estas áreas são exatamente aquilo no que a organização deve focar para melhorar o seu processo de desenvolvimento de software.

Para que uma empresa possa se qualificar em um determinado nível de maturidade do CMM, deve estar realizando os processos relacionados às áreas-chave daquele determinado nível. Todas as áreas-chave estão citadas na tabela abaixo:

Nível CMM	Foco	Áreas-chave de processo
1) Inicial	Pessoas competentes e heróis	
2) Repetível	Processos de gerenciamento de projetos	Gerenciamento de requisitos, Planejamento do projeto, Visão geral e acompanhamento do projeto, Gerenciamento de sub-contratados, Garantia da qualidade do software, Gerenciamento de configuração
3) Definido	Processos de engenharia	Foco do processo organizacional, Definição do processo organizacional, Programa de treinamento, Gerenciamento de software integrado, Engenharia de produto de software, Coordenação intergrupos, Revisão conjunta
4) Gerenciado	Qualidade do produto	Gerenciamento quantitativo dos processos, Gerenciamento da qualidade de software
5) Otimizado	Melhoramento contínuo do processo	Prevenção de defeitos, Gerenciamento de mudanças tecnológicas, Gerenciamento de mudanças no processo

5.4 - OBJETIVOS DAS ÁREAS-CHAVE DE PROCESSO

O modelo CMM define um conjunto de dois a quatro objetivos para cada área-chave.

Estes objetivos definem aquilo que deve ser alcançado no caso dos processos desta área-chave serem realmente realizados. Veja a seguir a lista destes objetivos.

5.4.1 - Repetível

- **Gerenciamento de requisitos**

- Os requisitos do sistema definidos para o software são controlados de forma a estabelecer um perfil mínimo a ser utilizado pela engenharia de software e pela administração.
- Os planos, produtos e atividades do software são sempre consistentes com os requisitos de sistema definidos para o software.

- **Planejamento do projeto**

- Estimativas relativas ao software são documentadas para uso no planejamento e acompanhamento do projeto do software.
- As atividades de projeto de software e compromissos assumidos são planejados e documentados.
- Grupos e pessoas afetadas concordam com seus compromissos relacionados ao projeto do software.

- **Visão geral e acompanhamento do projeto**

- Resultados reais são acompanhados de acordo do com o planejamento do software.
- Quando os resultados apresentam um significativo desvio do planejamento do software, são tomadas ações corretivas que são acompanhadas até o final do projeto.
- Mudanças nos compromissos assumidos são feitas em comum acordo com os grupos e indivíduos afetados.

- **Gerenciamento de sub-contratados**

- O contratante seleciona sub-contratos qualificados.
- O contratante e os sub-contratados estão de acordo no que diz respeito aos compromissos assumidos um com o outro.
- O contratante e os sub-contatados mantêm uma comunicação constante.
- O contratante acompanha os resultados reais do sub-contratado de acordo com os compromissos assumidos.

- **Garantia da qualidade do software**

- As atividades de garantia de qualidade de software são planejadas.
- A conformidade dos produtos de software e atividades com os padrões, procedimentos e requisitos é verificada objetivamente.
- Os grupos e indivíduos afetados são informados das atividades de garantia de qualidade de software e de seus resultados.
- Questões relacionadas à não conformidade que não são resolvidas dentro do projeto de software são encaminhadas à gerência geral.

5.4.2 - Definido

- **Gerenciamento de configuração**

- As atividades de gerenciamento de configuração são planejadas.
- Os produtos de trabalho de software são identificados, controlados e estão disponíveis.
- Mudanças nos produtos de trabalho identificados são controladas.

- **Foco do processo organizacional**

- São coordenadas atividades de desenvolvimento e melhoramento do processo de software em toda a organização.
- Os pontos fortes e fracos do processo de desenvolvimento de software utilizado são identificados, de acordo com um padrão de processo.
- São planejadas atividades de desenvolvimento e melhoramento do processo, a nível de organização.

- **Definição do processo organizacional**

- O processo padrão de desenvolvimento de software da organização é desenvolvido e mantido.
- A informação relacionada ao uso do processo padrão de desenvolvimento de software é coletada, revisada e disponibilizada.

- **Programa de treinamento**

- As atividades de treinamento são planejadas.
- É fornecido treinamento para o desenvolvimento de habilidades e conhecimentos necessários para realizar o gerenciamento do software e as funções técnicas.
- As pessoas no grupo de engenharia de software e outros grupos relacionados a software recebem o treinamento necessário para realizar as suas funções.

- **Gerenciamento de software integrado**

- O processo de software definido para o projeto é uma versão adaptada do processo padrão de desenvolvimento de software da organização.

- O projeto é planejado e gerenciado de acordo com o processo de desenvolvimento de software definido para o projeto.

- **Engenharia de produto de software**

- As atividades de engenharia de software são definidas, integradas e consistentemente realizadas para produzir o software.
- Os produtos de trabalho do software são mantidos consistentes entre si.

- **Coordenação intergrupos**

- Todos os grupos de trabalho afetados concordam com os requisitos dos clientes.
- Todos os grupos de trabalho afetados concordam com os acordos entre os grupos de engenharia.
- Os grupos de engenharia identificam, acompanham e resolvem todas as questões intergrupos.

- **Revisão conjunta**

- Atividade de revisão conjunta são planejadas.
- Defeitos nos produtos de trabalho são identificados e removidos.

5.4.3 - Gerenciado

- **Gerenciamento quantitativo dos processos**

- As atividades de gerenciamento quantitativo dos processos são planejadas.
- A performance do processo de desenvolvimento de software definido para o projeto é controlada quantitativamente.
- A capacidade do processo desenvolvimento de software padrão da organização é conhecida em termos quantitativos.

- **Gerenciamento da qualidade de software**

- As atividades de gerenciamento da qualidade de software do projeto são planejadas.
- Objetivos mensuráveis da qualidade do produto de software e suas prioridades são definidos.
- O progresso real em direção à realização dos objetivos de qualidade para os produtos de software é quantificado e gerenciado.

- **Prevenção de defeitos**

- As atividades de prevenção de defeitos são planejadas
- As causas comuns de defeitos são procuradas e identificadas.
- As causas comuns de defeitos são priorizadas e sistematicamente eliminadas.

5.4.4 - Otimizado

- **Gerenciamento de mudanças tecnológicas**

- A incorporação de mudanças tecnológicas é planejada.
- Novas tecnologias são avaliadas para determinar seu efeito na qualidade e na produtividade.
- Novas tecnologias adequadas são incorporadas na prática normal de toda a organização.

- **Gerenciamento de mudanças no processo**

- O melhoramento contínuo do processo é planejado.
- Toda a organização participa das atividades de melhoramento do processo de software.
- O padrão de processo de software da organização e os processos de software de cada projeto definido são melhorados continuamente.

5.5 CARACTERÍSTICAS COMUNS E PRÁTICAS-BASE

As características comuns são itens a serem observados para que se possa verificar a implementação e institucionalização de cada área-chave de processo. Elas podem indicar se a área-chave de processo é eficiente, repetível e duradoura.

São cinco as características comuns no modelo CMM e cada uma possui suas práticas-base a serem realizadas.

Característica Comum	Descrição	Práticas-base relacionadas
Compromisso de realizar	Atitudes a serem tomadas pela organização para garantir que o processo se estabeleça e seja duradouro.	Estabelecimento de políticas e apadrinhamento de um gerente experiente.
Capacidade de realizar	Pré-requisitos que devem existir no projeto ou na organização para implementar o processo de forma competente.	Alocação de recursos, definição da estrutura organizacional e de treinamento.
Atividades realizadas	Papéis e os procedimentos necessários para implementar uma área-chave de processo.	Estabelecimento de planos e procedimentos, realização do trabalho, acompanhamento do trabalho e tomada de ações corretivas, se necessário
Medições e análise	Necessidade de medir o processo e analisar as medições.	Realização de medições para determinar o estado e a efetividade das atividades realizadas.
Implementação com Verificação	Passos para garantir que as atividades são realizadas de acordo com o processo estabelecido.	Revisão, auditoria e garantia de qualidade.

As práticas-base descrevem as atividades que contribuem para atingir os objetivos de cada área-chave do processo. Em geral são descritas com frases simples, seguidas de descrições detalhadas (chamadas de sub-práticas) que podem até incluir exemplos.

As práticas-base devem descrever "o que" deve ser feito e não "como" os objetivos devem ser atingidos.

O modelo CMM inclui um extenso documento em separado, chamado "Práticas-base para o CMM", que lista todas as práticas-chave e sub-práticas para cada uma das áreas-chave de processo.

Em resumo, o CMM é definido em função de um conjunto de níveis de maturidade, áreas-chave de processo, características comuns e práticas-base .

5.7 - O CAMINHO PARA O NÍVEL 2 (REPETÍVEL)

Mais do que qualquer outro nível de maturidade, o nível repetível tem interesse especial, pois além do fato da grande maioria das empresas estar no nível 1 (Inicial), o caminho para o nível 2 é talvez o mais difícil de ser percorrido, pois trata-se de colocar o trem em movimento, enquanto a passagem para os níveis seguintes pode se beneficiar do "embalo" gerado pelos esforços de melhoria para se alcançar o nível 2.

O início do caminho para o nível 2 parte de um grande esforço para chamar a atenção para os problemas de software internamente. A partir de uma avaliação formal ou informal é necessário fazer com que todos dentro da empresa comprem a idéia da melhoria, de cima a baixo. Isto inclui a alta administração (diretoria), profissionais de informática (gerentes, analistas programadores, AD, DBA, produção, suporte etc. etc. etc.) e usuários (gerência e pessoal operacional).

Isto implica na necessidade de se constituir uma disciplina de comprometimento e é especialmente importante em relação ao tratamento de prazos. É evidente que a promessa de prazos fora da realidade é a principal causa de problemas de qualidade e insatisfação dos usuários. Mas não é o único problema. Assim, uma vez que se obtenha forte compromisso gerencial para as mudanças, pode-se iniciar a caminhada rumo ao nível 2.

O foco do nível 2 é o estabelecimento de procedimento básicos de Gerenciamento de Projetos. A falta de planejamento é a primeira coisa a ser atacada. Uma equipe que desenvolve seu trabalho aleatoriamente não poderá se beneficiar de técnicas de inspeção, metodologias de desenvolvimento ou métricas.

As diversas atividades de melhoria de software devem ser conduzidas por um grupo especialmente constituído para isto, normalmente chamado de SEPG (Software Engineering Process Group - Grupo de Processos de Engenharia de Software). A existência de um grupo especificamente encarregado destas tarefas é fundamental para garantir os projetos de melhoria. Obviamente, não são os desenvolvedores já pressionados que disporão de tempo para estas atividades.

Em termos de sua implementação, cada nível do CMM é composto de um certo número de ***Áreas-Chave de Processo (KPAs - Key Process Areas)***.

Estas áreas descrevem as questões que devem ser abordadas e resolvidas para se alcançar aquele nível. Assim, por exemplo, uma organização pode dominar uma KPA do nível três sem contudo estar sequer no nível dois em outras KPAs.

Em geral, considera-se que uma organização está em determinado nível caso todas as KPA definidas para aquele nível sejam cumpridas. Cada KPA, portanto, identifica um conjunto de atividades que, tomadas em conjunto, permitem alcançar uma série de objetivos importantes para se obter uma melhoria de processo.

As KPA do nível 2 são seis e são as seguintes:
- gerenciamento de requisitos;
- planejamento de projetos, acompanhamento e supervisão de projetos;
- gerenciamento de sub-contratação;
- garantia de qualidade de software; e
- gerenciamento de configuração.

5.7.1 - Gerenciamento de Requisitos

O propósito do Gerenciamento de Requisitos é estabelecer um entendimento comum entre o usuário e a equipe de projeto a respeito dos requerimentos que serão atendidos pelo projeto. Além disso, são estabelecidos procedimentos para a mudança dos requisitos, garantindo que os planos, cronogramas e recursos alocados estejam sempre coerentes com os requisitos.

Para cumprir esta KPA, a organização deve, em primeiro lugar, instituir procedimentos para a documentação dos requisitos. Estes requisitos devem também ser revisados por diversas áreas envolvidas (para verificar sua viabilidade) e ter sua qualidade controlada. Embora isto pareça óbvio, é impressionante ver a quantidade de gerentes de projeto que se comprometem com prazos antes de saber o que terão de fazer!

Em seguida, a organização precisa instituir procedimentos para a alteração dos requisitos. Isto significa que os requisitos não podem ser alterados "por telefone". A existência de um procedimento formal para alteração de requisitos tem dois benefícios:

- garante que os planos serão revistos para acomodar as alterações (isto é, renegociação de prazos e recursos); e
- funciona como um freio para alterações desnecessárias ou postergáveis (normalmente, a maioria), instituindo uma disciplina de controle de prioridades.

5.7.2 - Planejamento de Projetos

O propósito desta KPA é garantir que, a cada projeto, sejam elaborados planos razoáveis para o desenvolvimento. Isto envolve a realização de estimativas e o estabelecimento de compromissos.

Também se incluem aqui atividades de gerenciamento de riscos e alocação de recursos. Para cumprir esta KPA, a empresa deve instituir políticas e procedimentos padronizados para o planejamento de projetos.

Esta KPA também pressupõe que a empresa tenha definido um ciclo de vida de sistemas para ser usado no planejamento.

5.7.3 - Acompanhamento e Supervisão de Projetos

Planejar o projeto não basta. É preciso garantir, ao longo de sua execução, que os planos estão sendo cumpridos.

O propósito desta KPA é dar visibilidade sobre o progresso real do projeto em relação ao planejado, permitindo a tomada de ações corretivas quando a realidade estiver se afastando muito do que foi inicialmente planejado.

Para cumprir esta KPA, a organização deve instituir procedimentos para garantir que o progresso do projeto é continuamente comparado com o planejado, em termos de prazos, custos e qualidade. Além disso, os procedimentos devem estabelecer "gatilhos" a serem disparados quando houver muita distância, indicando as ações corretivas a serem tomadas em cada caso, incluindo alterações nos compromissos assumidos. A evidência, por exemplo, de que um prazo não será cumprido deve levar à renegociação de prazo, recursos, escopo ou nível de qualidade previamente acordados.

Esta KPA, naturalmente, inclui o acompanhamento dos fatores de risco identificados nas atividades de gerenciamento de riscos incluídas na KPA de Planejamento de Projetos.

Outro ponto importante coberto por esta KPA é o estabelecimento da obrigatoriedade da existência de pontos de revisão formal ao longo do projeto. Isto significa que, em determinados momentos do desenvolvimento, os produtos gerados até então são revisados formalmente pelas áreas envolvidas, como forma de garantia que o projeto está andando conforme deveria.

5.7.4 - Garantia de Qualidade de Software

O propósito desta KPA é prover visibilidade sobre a qualidade tanto dos processos utilizados pela equipe de projeto quanto dos produtos gerados. Isto inclui revisões formais e informais dos produtos e auditorias dos processos.

Tipicamente, em empresas com recursos suficientes, existe um grupo especialmente dedicado a estas tarefas. Quando estes recursos estão ausentes, pode-se executar as atividades desta KPA através de comitês de pessoas selecionadas nas áreas envolvidas, dedicando parte de seu tempo.

Sabendo-se que testes muito bem feitos (coisa muito rara, aliás) são capazes de remover em média apenas 60% dos defeitos de um programa, e que o custo da remoção de defeitos após a codificação é absurdamente maior do que se forem detectados e removidos em fases anteriores, fica evidente que a qualidade dos produtos do projeto deve ser acompanhada desde o início do desenvolvimento.

Certamente, se houvesse um procedimento de revisão da especificação, ao final da análise, a correção do problema seria infinitamente mais barata. Quanto à revisão dos processos, deve-se dizer que não basta revisar produtos.

Na maioria das vezes, produtos sem qualidade são resultados de processos sem qualidade. Não basta descobrir que um programa tem muitos "bugs". É necessário descobrir por quê estes "bugs" existem. Geralmente, a razão será algo como o não uso dos padrões definidos de programação, documentação de programação insuficiente, comunicação inadequada entre analistas e programadores etc.

A cada projeto, portanto, são desenvolvidos planos de garantia de qualidade, dentro do planejamento geral do projeto. Estes planos incluem pontos e procedimentos de verificação, tais como revisões de documentos, inspeções de código e, naturalmente, testes. O cumprimento destes planos permite a identificação de desvios de qualidade e, conseqüentemente, a adoção de ações corretivas o mais cedo possível.

5.7.5 - Gerenciamento de Configuração

O propósito desta KPA é garantir a integridade dos produtos gerados durante o ciclo de vida do projeto. Uma das maiores dores de cabeça em projetos complexos é gerenciar a configuração do software, incluindo aí o controle de versões. Em um determinado momento, um mesmo programa pode ter três ou mais versões. Além disso, pode haver confusão entre códigos-fonte e executáveis (como exemplo, o software livre Concurrent Version System – CVS em http://www.cvshome.org/).

Gerenciamento de Configuração é a KPA que evita que estes casos escabrosos ocorram.

Para cumprir esta KPA, a organização deve instituir procedimentos que garantam que as atividades de Gerenciamento de Configuração são planejadas, executadas e verificadas. Procedimentos de Controle de Mudanças são estabelecidos, garantindo que alterações em produtos (documentos, dados de teste, programas) são feitas de forma consistente e sob o conhecimento e aprovação de todos os envolvidos.

O uso de software de Controle de Configuração pode facilitar bastante a execução destas atividades. Também é necessária a criação de um grupo responsável pelas atividades de gerenciamento de configuração. Toda manutenção passa a ser considerada como a implantação de uma nova versão ou release do software.

5.8 – CONSIDERAÇÕES FINAIS SOBRE O CMM

É fácil entender por quê tão poucas organizações conseguem alcançar e se manter no nível 2.

Qualquer um que tenha alguma experiência na área sabe que a maioria das atividades acima ou são realizadas de forma aleatória, ou, na maioria das vezes, não são executadas de forma alguma. No entanto, são todas atividades necessárias para que exista um mínimo de ordem no desenvolvimento de software, e será difícil defender a tese de que é possível desenvolver software com qualidade na ausência de qualquer uma das atividades descritas.

Fica claro, portanto, que, em vez de burocracia sem sentido, padrões e processos bem definidos e usados podem fazer muito pela qualidade dos processos de desenvolvimento de uma empresa, aumentando a qualidade dos produtos e reduzindo prazos e custos por conta de muito menos retrabalho, que seria necessário como decorrência de um ambiente caótico.

Exemplo de Resultados da aplicação do CMM

Os dados da tabela abaixo se referem a empresas de software com número de desenvolvedores que varia entre 30 e 600 e as suas áreas de atuação incluem Telecomunicações, Sistemas embarcados em tempo real, Sistemas de Informação e Sistemas Operacionais.

Resultado	Média
Ganho anual de Produtividade	35 %
Ganho anual em detecção prévia de defeitos	22 %
Redução anual do tempo de entrega produto no mercado	19 %
Redução anual em relatos de defeitos no cliente	39 %
Retorno do Investimento	5.0

Fonte: Software Engineering Institute – JUN/2001

Unidade 6 – ISO 15504 (SPICE)
(**S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination)

6.1 - Introdução

O SPICE é uma norma de elaboração conjunta pela ISO e pelo IEC. Constitui-se de um padrão para a avaliação do processo de software, visando determinar a capacitação de uma organização. A norma visa, ainda, orientar a organização para uma melhoria contínua do processo. Ela cobre todos os aspectos da Qualidade do Processo de Software e representa o esforço conjunto de cinco centros técnicos espalhados pelo mundo (EUA, Canadá/América Latina, Europa, Pacífico Norte e Pacífico Sul).

Um grupo de estudos da ABNT está participa do processo de desenvolvimento, além de trabalhar na tradução das versões preliminares da norma para o português.

O SPICE inclui um modelo de referência, que serve de base para o processo de avaliação. Este modelo é um conjunto padronizado de processos fundamentais, que orientam para uma boa engenharia de software. O modelo é dividido em cinco grandes categorias de processo: Cliente-Fornecedor, Engenharia, Suporte, Gerência e Organização. Cada uma destas categorias é detalhada em processos mais específicos. Tudo isso é descrito em detalhes pela norma.

Além dos processos, o SPICE define também os 6 níveis de capacitação de cada processo (incompleto, executado, gerenciado, estabelecido, previsível e otimizado). O resultado de uma avaliação, portanto, um perfil da instituição em forma de matriz, onde temos os processos nas linhas e os níveis nas colunas.

6.2 - Categorias e Processos

Uma das contribuições do modelo SPICE é definir em seu modelo de referência todos os processos envolvidos no desenvolvimento de software, agrupados em categorias. Observe no quadro abaixo a estrutura completa das categorias, dos processos de cada categoria:

Processo	Descrição
CUS – Cliente-Fornecedor Processos que impactam diretamente os produtos e serviços de software na fornecedor para o cliente.	
CUS.1	Adquirir Software
CUS.2	Gerenciar necessidades do Cliente
CUS.3	Fornecer Software
CUS.4	Operar Software
CUS.5	Prover Serviço ao Cliente
ENG – Engenharia Processos que especificam, implementam ou mantém um sistema ou produto de software e sua documentação	
ENG.1	Desenvolver requisitos e o projeto do sistema
ENG.2	Desenvolver requisitos de software
ENG.3	Desenvolver o projeto do software
ENG.4	Implementar o projeto do software
ENG.5	Integrar e testar o software

ENG.6	Integrar e testar o sistema
ENG.7	Manter o sistema e o software
SUP – Suporte Processos que podem ser empregados por qualquer um dos outros processos	
SUP.1	Desenvolver a documentação
SUP.2	Desempenhar a gerência de configuração
SUP.3	Executar a garantia da qualidade
SUP.4	Executar a verificação dos produtos de trabalho
SUP.5	Executar a validação dos produtos de trabalho
SUP.6	Executar revisões conjuntas
SUP.7	Executar auditorias
SUP.8	Executar resolução de problemas
MAN – Gerência Processos que contém práticas de natureza genérica que podem ser usadas por quem gerencia projetos ou processos dentro de um ciclo de vida de software	
MAN.1	Gerenciar o projeto
MAN.2	Gerenciar a qualidade
MAN.3	Gerenciar riscos
MAN.4	Gerenciar subcontratantes
ORG – Organização Processos que estabelecem os objetivos de negócios da organização	
ORG.1	Construir o negócio
ORG.2	Definir o processo
ORG.3	Melhorar o processo
ORG.4	Prover recursos de treinamento
ORG.5	Prover infra-estrutura organizacional

A norma define detalhes de cada um dos processos mencionados acima. Para cada um deles existe uma definição mais detalhada, uma lista dos resultados da sua implementação bem sucedida e uma descrição detalhada de cada uma das práticas básicas.

6.3 - Níveis de Capacitação

O SPICE, entretanto, não se limita a listar categorias e processos. Seu principal objetivo, na realidade, é avaliar a capacitação da organização em cada processo e permitir a sua melhoria. O modelo de referência do SPICE inclui seis níveis de capacitação. Cada um dos processos mencionados acima deve ser classificado nestes níveis. Os níveis são descritos a seguir:

Nível	Nome	Descrição
0	Incompleto	Há uma falha geral em realizar o objetivo do processo. Não existem produtos de trabalho nem saídas do processo facilmente identificáveis.
1	Realizado ou Executado	O objetivo do processo em geral é atingido, embora não necessariamente de forma planejada e controlada. Há um consenso na organização de que as ações devem ser realizadas e quando são necessárias. Existem produtos de trabalho para o processo e eles são utilizados para atestar o atendimento dos objetivos.
2	Gerenciado	O processo produz os produtos de trabalho com qualidade aceitável e dentro do prazo. Isto é feito de forma planejada e controlada. Os produtos de trabalho estão de acordo com padrões e requisitos.
3	Estabelecido	O processo é realizado e gerenciado usando um processo definido, baseado em princípios de Engenharia de Software. As pessoas que implementam o processo usam processos aprovados, que são versões adaptadas do processo padrão documentado.
4	Previsível	O processo é realizado de forma consistente, dentro dos limites de controle, para atingir os objetivos. Medidas da realização do processo são coletadas e analisadas. Isto leva a um entendimento quantitativo da capacitação do processo a uma habilidade de predizer a realização.
5	Otimizado	A realização do processo é otimizada para atender às necessidade atuais e futuras do negócio. O processo atinge seus objetivos de negócio e consegue ser repetido. São estabelecidos objetivos quantitativos de eficácia e eficiência para o processo, segundo os objetivos da organização. A monitoração constante do processo segundo estes objetivos é conseguida obtendo feedback quantitativo e o melhoramento é conseguido pela análise dos resultados. A otimização do processo envolve o uso piloto de idéias e tecnologias inovadoras, além da mudança de processos ineficientes para atingir os objetivos definidos.

6.4 - Os 9 manuais do SPICE

Esta norma é composta por um conjunto de 9 manuais, totalizando quase 400 páginas, conforme o detalhamento a seguir:

Parte	Páginas	Descrição
1	16	Guia de Introdução e Conceitos
2	38	Modelo de referência para processos e capacidade de processos
3	7	Realizando uma avaliação
4	36	Guia para realização de uma avaliação
5	145	Um modelo de avaliação e guia de indicadores
6	31	Guia para qualificação de avaliadores
7	47	Guia para uso no melhoramento de processos
8	25	Guia para uso na determinação da capacidade do processo de fornecedor
9	9	Vocabulário

Unidade 7 – PSP - Personal Software Process

(extraído de Rezende, Denis A. – Engenharia de Software e Sistemas de Informação – 2ed.- Rio de Janeiro: Brasport, 2002)

O PSP (Processo Pessoal de Software) é um processo usado por engenheiros de software individuais, projetado para controlar, gerenciar e melhorar o seu próprio trabalho.

Este processo foi desenvolvido por Watts S. Humphrey e relatado em sua obra com intenção de orientar como planejar e desenvolver módulos de software ou pequenos programas, podendo ser adaptado para outras tarefas pessoais.

O PSP assim como o CMM, é baseado no princípio da melhoria contínua de processos. O foco principal do modelo CMM é a melhoria da capacidade organizacional das empresas desenvolvedoras de solução; já o modelo PSP está focado no trabalho individual dos engenheiros de software.

A proposta do modelo PSP é fornecer um método de trabalho que ensine processos de software aos próprios desenvolvedores, partindo do princípio do envolvimento pessoal no próprio processo.

O modelo PSP é baseado nas mesmas práticas industriais encontradas no CMM da SEI, só que em uma escala menor para o uso individual dos engenheiros de software, criando módulos (rotinas) ou pequenos programas com qualidade.

Como o modelo do PSP é flexível, também pode ser adaptado para atender desde a especificação dos requisitos funcionais que o software pretende atender até os seus respectivos testes, ou seja, o projeto de sistemas como um todo.

Com relação aos processos, o modelo PSP os trata como uma seqüência de passos requeridos para desenvolvimento ou manutenção de software e a definição de processos de software é a descrição destes processos.

Para o sucesso do modelo PSP, dois itens fundamentais são necessários: capacitação (excelência) pessoal e motivação pessoal.

Os principais objetivos do modelo PSP são:

- melhorar as estimativas, planejamentos e respectivos acompanhamentos;
- criar um comprometimento pessoal para a qualidade;
- envolvimento total do engenheiro de software na melhoria contínua e constante do processo de software; e
- organizar o backlog (pendências) de atividades, evitando o excesso de comprometimentos.

E na medida da melhoria individual o engenheiro de software, a conseqüência é a melhoria da capacidade de empresa como um todo. E neste sentido, as empresas podem estimar melhor os custos do software, com melhor gestão no planejamento de desenvolvimento de soluções e na qualidade quanto à facilidade de encontrar e remover defeitos antes da compilação, testes e verificação de software.

Ainda, o tempo do ciclo de desenvolvimento de software é diminuído pelo planejamento melhor elaborado e pela eliminação de retrabalhos, gerando produtos mais efetivos.

7.1 – Níveis do PSP

Os níveis do modelo estão divididos em 7 estágios incrementais. Os superiores adicionam características aos níveis já implantados, minimizando o impacto da mudança no processo do engenheiro de software. Será necessário, apenas, a adaptação 'as novas técnicas já implantadas e conhecidas. Assim estão distribuídos:

- PSP 0 – Ponto de partida do processo ou medição pessoal;
  - processo atual e as medidas pessoais básicas (tempo, tamanho e defeitos);
  - PSP 0.1 – padrões de codificação, proposta de melhoria e estimativas.

- PSP 1 – Processo de planejamento pessoal;
  - estimativas e relatórios de teste;
  - PSP 1.1 – planejamento de tarefas e de tempo(cronograma).

- PSP 2 – Gestão da qualidade pessoal;
  - revisão de códigos e projeto;
  - PSP 2.1 – modelos ou padrões de projetos.

- PSP 3 – Processo pessoal cíclico.
  - desenvolvimento cíclico;

7.2 – PSP 0: Ponto de Partida do Processo ou Medição Pessoal

PSP 0 e PSP 0.1 fornecem a iniciação ao modelo PSP e estabelecem o início de uma base histórica de tamanho, tempo e informações sobre defeitos.

Neste nível, os engenheiros de software elaboram os formulários de Planejamento, Tempo, Defeitos e a Proposta de Melhoria de Processos (PIP) e demais procedimentos propostos para o trabalho pessoal. Estes formulários formarão a base histórica para os outros níveis do PSP.

Neste nível (PSP 0), um modelo em cascata de 6 fases pode ser utilizado:

1 – Planejamento → Planejamento do trabalho e a documentação deste planejamento;
2 – Projeto → Projeto do Programa (ou módulo);
3 – Codificação → Implementação do Projeto;
4 – Compilação → Compilação do programa e correção dos encontrados;
5 – Teste → Teste do programa e correção dos erros encontrados;
6 – Finalização → Registro dos tempos, defeitos e dados sobre o tamanho na documentação do planejamento.

Os defeitos são tratados como quaisquer mudanças que devem ser elaboradas no projeto ou na codificação dos programas, para que ele seja compilado e testado sem problemas.

A Proposta de Melhoria de Processos (PIP) que aparece no PSP 0.1 é um formulário que possibilita o registro de problemas, experiência e sugestões de soluções e melhorias.

Ainda no PSP 0 são apresentadas as *métricas derivadas* ou medidas decorridas das três medidas básicas do PSP (tempo de desenvolvimento, defeitos e tamanho) para auxiliar o engenheiro de software a melhorar a gestão do projeto e o seu desempenho de trabalho.

Como exemplo das medidas, podemos citar:

- *tempo de interrupção*: intervalo de tempo de pequenas interrupções na execução do projeto, tais como chamadas telefônicas, cafezinho, sanitários, etc.;
- *tempo decorrido:* intervalo de tempo do início a parada sem interrupções (em minutos);
- *tempo planejado na fase*: tempo estimado despendido para um projeto;
- *tempo real da fase:* soma dos tempos decorridos para uma fase de um projeto;
- *tempo total:* soma do planejado ou a soma do tempo real para todas as fases de um projeto;
- *tempo atual de fases:* soma dos tempos reais até o momento;
- *tempo total atual:* soma dos tempos atuais de fases para todas as fases de todos os projetos;
- *tempo de compilação:* tempo entre a primeira compilação até a primeira compilação sem erros;
- *tempo de teste:* tempo entre o primeiro teste até todos os testes corretos;
- *defeito:* qualquer elemento do projeto do programa ou implementação que deve ser trocado para corrigir programas;
- *tipo de defeito:* conforme tabela;
- *tempo de resolução:* tempo para encontrar e resolver um defeito;
- *tempo de avaliação:* tempo despendido no projeto e revisão do código;
- *tempo de falhas:* tempo despendido na compilação e testes;

7.3 – PSP 1 e PSP 1.1 – Processo de Planejamento Pessoal

Relatam sobre as técnicas de gestão do projeto pessoal, compreendendo as estimativas de tamanho e esforço, planejamento e acompanhamento de cronogramas. As estimativas e relatórios de testes são introduzidos no PSP 1, enquanto o planejamento e o acompanhamento de tarefas e tempo são introduzidos no PSP 1.1.

Assim, o engenheiro de software assimila a:

- relação entre o tamanho dos programas que desenvolve e o tempo de desenvolvimento (esforço);
- compreensão dos objetivos que possam ser atendidos;
- preparação de um plano ordenado para realizar seu trabalho; e
- o estabelecimento de uma base para avaliar seu trabalho.

A importância dessas técnicas para grandes projetos de sistemas é entendida, porém poucos engenheiros de software aplicam estas técnicas em seu próprio trabalho individual.

Mesmo que as estimativas não sejam ainda exatas, a intenção neste nível é de aprender a fazer estimativas precisas, e na medida que a base de dados aumenta, esses erros tendem a diminuir.

7.4 – PSP 2 e PSP 2.1 – Gestão da Qualidade Pessoal

Tem como objetivo auxiliar o engenheiro de software a trabalhar de forma real e objetiva os defeitos que ele próprio insere no programa.

Estes níveis adicionam métodos para gestão da qualidade ao PSP, ou seja, o projeto pessoal, a revisão de códigos, o projeto com notação padronizada, a padronização de

projeto, as técnicas de verificação de projeto e as medidas para gestão do processo e qualidade do produto.

Tendo em vista que a maioria dos erros de software são geralmente problemas de sintaxe e outros pequenos e ingênuos erros, levam o engenheiro a pensar que pode melhorar o processo somente pelo trabalho de tentativas de execuções contínuas de acerto dos códigos, compilação, correção.

Definitivamente a prevenção de defeitos é fundamental e necessária; neste sentido o PSP 2 apresenta conceitos através da gestão de defeitos, com relatórios de contagem de erros anteriores (PSP 0), construindo e usando listas de verificação (*check lists*) para a revisão de projeto e de código.

O principal objetivo deste nível é localizar, evitar e remover todos os defeitos antes da primeira compilação, associando a esta medida o conceito de ganho (*yield*), que é definido como o percentual de defeitos eliminado antes da compilação.

A revisão de projeto e a revisão de código são introduzidos no processo para auxiliar o engenheiro de software a alcançar 100% de ganho. As revisões pessoais são conduzidas individualmente em seu projeto e código. Estas revisões são estruturadas, o processo de revisão é dirigido aos dados, sendo guiadas por listas de verificação de revisão pessoal derivadas dos dados históricos dos defeitos do engenheiro de software.

A utilização de dados históricos para planejamento da qualidade e do controle da qualidade durante a construção do software remove os defeitos por ele inseridos antes da primeira compilação. Assim, pode ser estimulado o número de defeitos possíveis de remoção, usando a correlação histórica entre a taxa de revisão e o ganho para um planejamento efetivo, gerando a revisão efetiva, controlando a qualidade por monitoração dos defeitos realmente inseridos e removidos em relação ao planejado, considerando a taxa de revisão como o limite estabelecido.

Com a auto-avaliação é possível melhorar a própria performance, de forma gradual, em cada processo, aplicando o PSP 2 e revisando os padrões, focando a perfeição.

A atividade de revisão ainda é a melhor forma de corrigir erros de compilação e muito dos erros encontrados na fase de teste, sendo necessária a melhoria da qualidade do projeto. O PSP 2.1 é dirigido a essa necessidade, pois adiciona notação de projeto, padrões de projeto e método para verificação do projeto. Desta forma se assegura que o engenheiro de software examine e documente o projeto para diferentes perspectivas, melhorando o processo de projeto e verificando o projeto com as revisões efetivas.

Os padrões de projeto fornecem quatro perspectivas:

- especificação operacional;
- especificação funcional;
- especificação de estados; e
- especificação lógica.

7.5 – PSP 3 – Desenvolvimento Cíclico do Processo

É o último nível do PSP, introduzindo métodos para engenheiros de software desenvolverem projetos maiores.

Assim, os problemas de comunicação e de coordenação, que são relevantes em projetos de larga escala, não são abordados no PSP, caso em que se deve optar pela utilização do modelo CMM ou TSP (*Team Software Process*), por exemplo.

Este nível é voltado para a utilização do PSP em grandes projetos, sem prejudicar a qualidade e a produtividade, onde o engenheiro de software concebe que sua produtividade é a máxima dentro de faixas de valor de tamanho do programa. Abaixo destas faixas, a produtividade tende a cair pelo excesso de documentação de programas e, acima, cai por causa da complexidade de gestionar grandes programas.

Este nível do PSP introduz limites de tamanho e estratégia de desenvolvimento cíclico, onde programas grandes são divididos em partes para o desenvolvimento e a integração, fazendo com que o engenheiro de software trabalhe com o máximo de sua produtividade elevando, em conseqüência, a qualidade do produto.

Para isso, dois novos formulários são criados: sumário do ciclo para registrar tamanho, tempo de desenvolvimento e defeitos de cada ciclo e relatório de conseqüências que registra o que pode afetar o módulo no futuro ou na conclusão dos ciclos.

Quando o engenheiro de software utiliza o PSP 3, ele decompõe seu projeto em uma série de ciclos PSP 2.1, integrando e testando a saída de cada ciclo. Sendo que os programas produzidos com o PSP 2.1 são de efetiva qualidade, os custos de teste e integração são minimizados. Evidentemente que, a cada interação de ciclo, todas as etapas do PSP 2.1 devem ser elaboradas.

O PSP 3 torna-se efetivo se os módulos de cada ciclo forem efetivamente de alta qualidade, onde os engenheiros de software podem se concentrar no ciclo atual de desenvolvimento, não se preocupando mais com os defeitos existentes nos primeiros ciclos, carecendo das revisões de código e do projeto dos níveis anteriores.

Finalizando, o modelo PSP é uma ferramenta que pode ser aplicada em qualquer empresa e em qualquer projeto de software, visando a melhoria contínua da qualidade de desenvolvimento pessoal dos engenheiros de software, da empresa como um todo e de seus respectivos produtos de tecnologia.

Unidade 8 – Medidas da Qualidade de Software

A principal meta da engenharia de software é produzir um sistema, aplicação ou produto de alta qualidade. A qualidade de um sistema, aplicação ou produto não é melhor que os requisitos que descrevem o problema, o projeto que modela a solução, o código que leva a um programa executável e os testes que exercitam o software para descobrir erros.

8.1 – Medição de Qualidade

Apesar das várias medidas de qualidade de software, a correção, a manutenibilidade, a integridade e a utilização fornecem indicadores úteis a equipe de software. A seguir, sugerimos definições e medidas para cada uma.

- **Correção.** Um programa precisa operar corretamente ou é de pouco valor para seus usuários. Correção é o grau em que o software desempenha sua necessária função. A medida mais comum de correção é *defeitos por KLOC*, onde defeito é definido como uma falha verificada de obediência aos requisitos. Quando se considera a qualidade geral de um produto de software, defeitos são problemas relatados por um usuário do programa, após o programa ter sido entregue para uso geral. Para fins de avaliação de qualidade, defeitos são contados durante o período de um ano.

- **Manutenibilidade.** Manutenção de software consome mais esforço do que qualquer outra atividade de engenharia de software. Manutenibilidade é a facilidade com que um programa pode ser corrigido, se um erro é encontrado, adaptado, se seu ambiente se modifica, ou aperfeiçoado, se o cliente deseja uma modificação nos requisitos. Não há modo de medir manutenibilidade diretamente; assim precisamos usar medidas indiretas. Uma métrica simples orientada a tempo é tempo médio para modificação (*mean-time-to-change*, MTTC), que é o tempo despendido para analisar o pedido de modificação, projetar uma modificação adequada, implementar a modificação, testá-la e distribuí-la para todos os usuários. Em média, programas manuteníveis vão ter um MTTC mais baixo (para tipos equivalentes de modificação) do que programas não-manuteníveis.

- **Integridade.** A integridade de software tornou-se cada vez mais importante na era dos hackers e firewalls. Esse atributo mede a capacidade do sistema de resistir a ataques (tanto acidentais quanto intencionais) à sua segurança. Os ataques podem ser feitos aos três componentes do software: programas, dados e documentos. Para medir a integridade, dois atributos adicionais precisam ser definidos: ameaça e segurança. *Ameaça* é a probabilidade (que pode ser estimada ou originada de experiência empírica) de um ataque, específico, ocorrer dentro de um certo período. *Segurança* é a probabilidade (que, também, pode ser estimada ou originada de experiência empírica) de um ataque ser repelido. A integridade do sistema pode então ser assim definida (ameaça e segurança devem ser somadas por tipo de ataque):

$$\text{INTEGRIDADE} = \text{SOMATÓRIO}\ [(1 - \text{AMEAÇA}) \times (1 - \text{SEGURANÇA})]$$

- **Utilização.** A expressão "uso amigável" tornou-se onipresente na discussão de produtos de software. Se um programa não é de uso amigável, freqüentemente, está destinado ao fracasso, mesmo se as funções que ele desempenha forem valiosas. Utilização é uma tentativa de quantificar a característica de ser amigável ao uso e pode ser medida em função de quatro tópicos:

  - a aptidão física e/ou intelectual necessária para aprender a lidar com o sistema;
  - o tempo necessário para se tornar moderadamente eficiente no uso do sistema;
  - o aumento líquido de produtividade (relativo a abordagem que o sistema substitui), medido quando o sistema é usado por alguém moderadamente eficiente; e
  - uma avaliação subjetiva (algumas vezes obtida por intermédio de um questionário) das atitudes dos usuários com relação ao sistema.

8.2 – Eficiência na remoção de defeitos

Uma métrica de qualidade, que fornece benefícios tanto para projetos quanto para processos, é a eficiência na remoção defeitos (*defect removal efficiency*, DRE). A DRE é uma medida da capacidade de filtragem das atividades de controle e garantia de qualidade de software, à medida que são aplicadas as atividades de arcabouço de processo. Quando considerada para o projeto todo, a DRE é definida da seguinte maneira:

$$\mathbf{DRE = E / (E + D)}$$

em que, E é a quantidade de erros encontrados antes da entrega do software ao usuário final e D é a quantidade de defeitos encontrados após a entrega.

O valor ideal da DRE é 1. Isto é, nenhum defeito é encontrado no software. À medida que E aumenta (para um dado valor de D), o valor da DRE se aproxima de 1. De fato, à medida que E aumenta, é provável que o valor final de D diminua (os erros são filtrados antes que se transformem em defeitos). Se usada como métrica, que fornece indicador da capacidade de filtragem das atividades de garantia e controle de qualidade, a DRE motiva uma equipe de projeto de software a instituir técnicas para encontrar tantos erros quanto possível antes da entrega.

As medidas de qualidade de software requerem fatores de qualidade de software. Estes focalizam três aspectos importantes de um software: suas características operacionais, sua manutenibilidade em relação a mudanças e sua adaptabilidade a novos ambientes. Estes aspectos são apresentados de acordo com as seguintes descrições:

- confiabilidade: medida que se pode esperar que um programa execute sua função pretendida com a precisão exigida;
- correitude: medida que um programa satisfaz sua especificação e cumpre os objetivos visados pelo cliente;
- eficiência: quantidade de recursos de computação e código exigida para que um programa execute sua função;
- flexibilidade: esforço exigido para modificar um programa operacional;
- integridade: medida que o acesso ao software ou a dados por pessoas não autorizadas pode ser controlado;
- interoperabilidade: esforço exigido para acoplar (unir ou ligar, completando ou não) um sistema ao outro;
- manutenibilidade: esforço exigido para localizar e reparar erros num programa;
- portabilidade: esforço exigido para transferir o programa de um ambiente de sistema de hardware e/ou de software para outro;
- reusabilidade: medida que um programa (ou partes de um programa) pode ser reusado em outras aplicações – relacionadas ao empacotamento e escopo das funções que o programa executa;
- testabilidade: esforço exigido para testar um programa a fim de garantir que ele execute sua função pretendida;
- usabilidade: esforço para aprender, operar, preparar a entrada e interpretar a saída de um programa.

As métricas de software e as medidas de qualidade de software devem ser trabalhadas junto com uma metodologia de desenvolvimento de software e também devem constar nas normas e padrões técnico-operacionais da empresa.