Engenharia de Software

Qualidade de Software

Conteúdo

- Introdução
- Conceitos de Qualidade
- Fatores de Qualidade de Software
- Métricas de Qualidade
- Controle e Garantia de Qualidade de Software
- Certificação de Qualidade
- Conclusão

Introdução

- O principal objetivo da Engenharia de Software (ES) é ajudar a produzir software de qualidade;
- Empresas que desenvolvem software de qualidade são mais competitivas;
- Empresas que utilizam software de alta qualidade podem, em geral, oferecer um melhor serviço a um preço mais competivo.

Conceitos de Qualidade

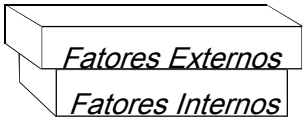
- Definição genérica:
  - "Propriedade, atributo ou condição das coisas ou das pessoas capaz de distingui-las das outras e de lhes determinar a natureza" (Aurélio).
- Outras definições:
  - Qualidade é estar em conformidade com os requisitos dos clientes;
  - Qualidade é antecipar e satisfazer os desejos dos clientes;
  - Qualidade é escrever tudo o que se deve fazer e fazer tudo o que foi escrito.

Conceitos de Qualidade

- Segundo a atual norma brasileira sobre o assunto (NBR ISO 8402), qualidade é:
  - A totalidade das características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.
- Definição de qualidade de software:
  - "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 profissionalmente desenvolvido" (Pressman).

Fatores de Qualidade de Software

- A noção de qualidade de software pode ser descrita por um grupo de fatores, requisitos ou atributos, tais como: confiabilidade, eficiência, facilidade de uso, modularidade, legibilidade, etc;
- Podemos classificar estes fatores em dois tipos principais: externos e internos;

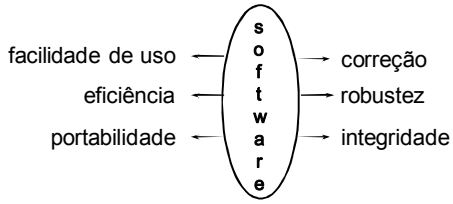


Fatores de Qualidade de Software

- Fatores externos são percebidos tanto pelas pessoas que desenvolvem software quanto pelos usuários.
  - Por exemplo, confiabilidade, eficiência e facilidade de uso são fatores externos;
- Fatores internos são percebidos apenas pelas pessoas que desenvolvem software.
  - Por exemplo, modularidade e legibilidade são fatores internos;

Se os fatores internos forem observados, os fatores externos serão conseqüentemente observados. De fato, os fatores internos são um meio para se alcançar os fatores externos.

Fatores Externos de Qualidade de Software



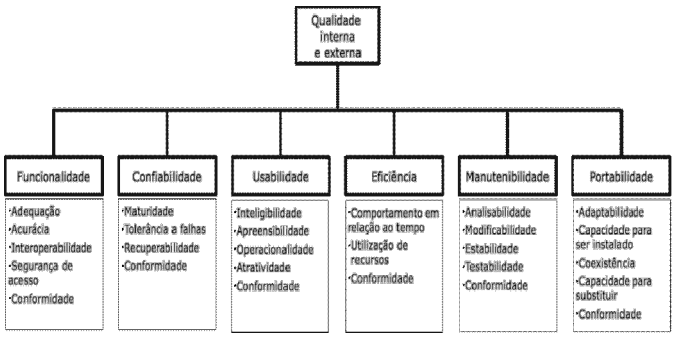
Fatores Externos de Qualidade de Software

- Facilidade de uso: a facilidade de aprender como usar o software;
- Eficiência: o bom uso dos recursos computacionais;
- Portabilidade: a facilidade de transferir software entre ambientes operacionais.

Fatores Externos de Qualidade de Software

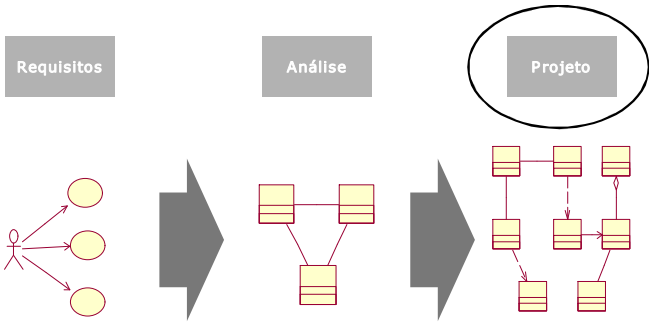
- Correção: habilidade do software executar suas tarefas exatamente como definida pelos requisitos e especificação*;*
- Robustez: habilidade de um software funcionar mesmo em condições anormais;
- Integridade: a habilidade do sistema de proteger seus vários componentes contra acessos ou modificações indevidos.

Modelo de Qualidade da Norma ISO 9126



Entendendo REQUISITOS

Etapas de um projeto



Elicitação de requisitos e análise

- Esta atividade divide-se em dois esforços maiores:
  - Elicitação dos requisitos em si
    - Técnicas de elicitação
  - Análise do que foi elicitado
    - Processo de análise

Que é um requisito?

- Tanto pode ser
  - Uma declaração abstrata de alto nível de um serviço
  - Como uma restrição do sistema
- Quanto uma especificação funcional matemática detalhada

Elicitação de Requisitos

- Também denominada de descoberta de requisitos
- Envolve pessoal objetivando descobrir o domínio de aplicação, serviços que devem ser fornecidos bem como restrições
- Deve envolver usuários finais, gerentes, pessoal envolvido na manutenção, especialistas no domínio, etc. (*Stakeholders*).

Visão dos Requisitos

- Requisitos do Usuário
  - Declarações em linguagem natural com diagramas de serviços que o sistema deve oferecer e suas restrições operacionais. Escrito para os clientes
- Requisitos do Sistema
  - Documento estruturado com descrições detalhadas sobre os serviços do sistema. Contrato entre cliente e fornecedor

Tipos de Requisitos

- Requisitos Funcionais
- Requisitos Não-Funcionais
- Requisitos de Domínio

Requisitos Funcionais

- Descreve funcionalidade e serviços do sistema
- Depende do tipo do software, usuários esperados e o tipo do sistema onde o software é usado
- Requisitos funcionais do usuário devem ser declarações de alto nível sobre o que o sistema deve fazer
- Requisitos funcionais do sistema devem descrever os serviços do sistema em detalhes

Exemplos de R.F.

- [RF001] Usuário pode pesquisar todo ou um sub-conjunto do banco de dados
- [RF002] Sistema deve oferecer visualizadores apropriados para o usuário ler documentos armazenados
- [RF003] A todo pedido deve ser associado um identificador único (PID), o qual o usuário pode copiar para a área de armazenamento permanente da conta

Imprecisão dos Requisitos

- Problemas podem surgir quando os requisitos não são estabelecidos precisamente
- Requisitos ambíguos podem ser interpretados diferentemente por usuários e desenvolvedores
- Considere o termo 'visualizadores apropriados´
  - Visualizador de propósito especial para cada tipo de documento diferente (Usuário)
  - Oferecer um visualizador de texto que mostra o conteúdo do documento (Desenvolvedor)

Completude e consistência dos Requisitos

- Em princípio, requisitos devem ser completos e consistentes
- Completo
  - Descrições de todos os serviços
- Consistência
  - Não deve haver conflitos e contradições nas descrições dos serviços
- Na prática, torna-se impossível produzir um documento de requisitos completo e consistente

Exercício

- Dê alguns exemplos de R.F.s para:
  - 1. Sistema da padaria de pequeno porte;
  - 2. Sistema de locadora.

Requisitos Não-Funcionais

- Definem propriedades e restrições do sistema (tempo, espaço, etc)
- Requisitos de processo também podem especificar o uso de determinadas linguagens de programação, método de desenvolvimento
- Requisitos não-funcionais podem ser mais críticos que requisitos funcionais. Não satisfaz, sistema inútil.

Requisitos Não-Funcionais

- Definem propriedades e restrições do sistema (tempo, espaço, etc)
- Requisitos de processo também podem especificar o uso de determinadas linguagens de programação, método de desenvolvimento
- Requisitos não-funcionais podem ser mais críticos que requisitos funcionais.

Requisitos Não-Funcionais

- Devido à sua própria definição, requisitos não-funcionais são esperados mensuráveis
- Assim, deve-se associar forma de medida/referência a cada requisito não-funcional elicitado

Medidas de Requisitos (Não-Funcionais)

Propriedade	Medida
Velocidade	Transações processadas/seg Tempo de resposta do usuário/evento
Tamanho	K bytes Nº de chips de RAM
Facilidade de uso	Tempo de treinamento Nº de quadros de ajuda
Confiabilidade	Tempo médio de falhas Probabilidade de indisponibilidade Taxa de ocorrência de falhas
Robustez	Tempo de reinício após falha Percentual de eventos causando falhas Probabilidade de corrupção de dados após falha
Portabilidade	Percentual de declarações dependentes do destino Nº de sistemas destino

Classificação de R. N. F.

- Requisitos do Produto
  - Produto deve comportar-se de forma particular (velocidade de execução, confiabilidade, etc.)
- Requisitos Organizacionais
  - Conseqüência de políticas e procedimentos organizacionais (padrões de processo usados, requisitos de implementação, etc.)
- Requisitos Externos
  - Conseqüência de fatores externos ao sistema e ao processo de desenvolvimento (legislação, etc.)

Exemplos de R. N. F.

- Requisitos do Produto
  - [RNF001] Toda consulta ao B.D., baseada em código de barras, deve resultar em até 5 s
- Requisitos Organizacionais
  - [RNF002] Todos os documentos entregues devem seguir o padrão de relatórios XYZ-00
- Requisitos Externos
  - [RNF003] Informações pessoais do usuário não devem ser vistas pelos operadores do sistema

Objetivos e Requisitos

- Requisitos não-funcionais podem ser muito difíceis de serem estabelecidos precisamente e requisitos imprecisos difíceis de verificar
- Objetivo
  - Intenção geral do usuário (facilidade de uso)
- Requisito não-funcional verificável
  - Declaração usando alguma medida que possa ser testada objetivamente
- Objetivos são úteis aos desenvolvedores uma vez que eles apresentam as intenções dos usuários do sistema

Objetivos e Requisitos (Exemplos)

- Um objetivo do sistema
  - Sistema deve ser fácil de usar por usuários experientes e organizado de forma a minimizar erros do usuário
- Um requisito não-funcional verificável
  - Usuários experientes devem ser capazes de usar toda a funcionalidade do sistema após 2h de treinamento. Após treinamento, erros não podem ultrapassar 2 por dia

Exercício

- Dê alguns exemplos de R.N.F.s para:
  - 1. Sistema da padaria de pequeno porte;
  - 2. Sistema de locadora.

Requisitos de Domínio

- Derivados do domínio da aplicação e descrevem características do sistema e qualidades que refletem o domínio
- Podem ser requisitos funcionais novos, restrições sobre requisitos existentes ou computações específicas
- Se requisitos de domínio não forem satisfeitos, o sistema pode tornar-se não prático

Requisitos de Domínio (Problemas)

- Entendimento
  - Não é entendido pelos engenheiros de software que vão desenvolver a aplicação
- Implicitude
  - Especialistas no domínio entendem a área tão bem que não tornam todos os requisitos de domínio explícitos

Requisitos de Domínio (Exemplo 1)

- A desaceleração do trem deve ser computada através da fórmula

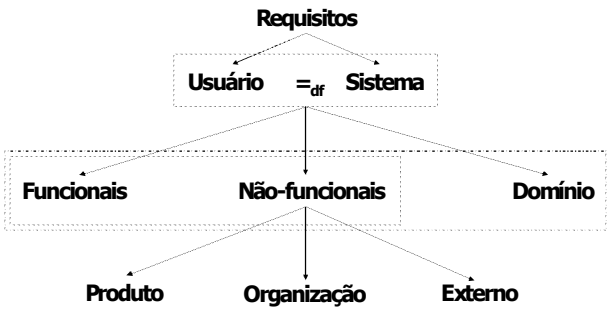
  $D_{trem}=D_{controle}+D_{gradiente}$

  onde ...

Exercício

- Dê alguns exemplos de domínio para:
  - 1. Sistema da padaria de pequeno porte;
  - 2. Sistema de locadora.

Requisitos



Requisitos do Usuário

- Devem descrever requisitos funcionais e não-funcionais de tal forma que sejam entendíveis pelos usuários do sistema que não têm conhecimento técnico detalhado
- Requisitos do usuário são definidos usando linguagem natural, tabelas e diagramas

O Documento de Requisitos

- O documento de requisitos é a declaração oficial do que é requerido dos desenvolvedores do sistema
- Deve incluir uma definição e uma especificação dos requisitos
- Deve declarar o que o sistema deve fazer e não como

Guias para Escrever Requisitos

- Adote um formato padrão e use-o para todos os requisitos
- Use linguagem consistentemente. Use **deverá** para requisitos obrigatórios e **deveria** para os desejáveis
- Use **texto em negrito** para identificar partes chave dos requisitos
- Evite jargão de computação

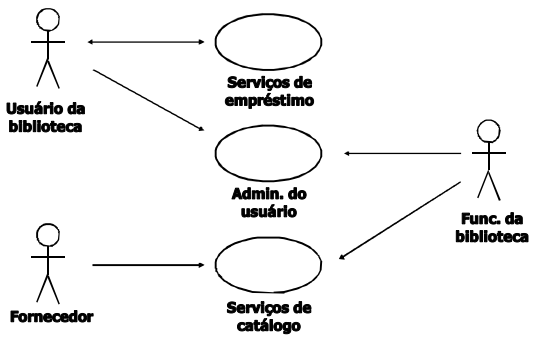
Requisitos do Sistema

- Especificações mais detalhadas dos requisitos do usuário
- Serve de base para projetar o sistema
- Pode ser usado como parte do contrato do sistema
- Geralmente expressos usando modelos do sistema
  - UML

Alternativas para Especificação

Notação	Descrição
Linguagem natural estruturada	Depende de formulários padrão
Linguagem de descrição de projeto	Linguagem semelhante a ling. prog. mas com recursos abstratos para especificar requisitos
Notações gráficas	Ling. gráfica com anotações de texto (Use-cases de UML)
Especificações Formais	Ling. com sintaxe e semântica bem-definidas (OBJ, Z, CSP, LOTOS)

UML (Use-Case)



CSP

CaixaAut = cartao → (saldo → (CC → ...
□ Poup→ ...)
□ extrato → ...
□ ...)

Estrutura de um Documento de Requisitos

- Introdução
- Glossário
- Definição dos Requisitos do Usuário
- Arquitetura do Sistema
- Especificação dos Requisitos do Sistema
- Modelos do Sistema
- Evolução do Sistema
- Apêndices
- Índice

Bibliografia

- Sommerville, I. Software Engineering
- Kruchten, P. The Rational Unified Process: An Introduction

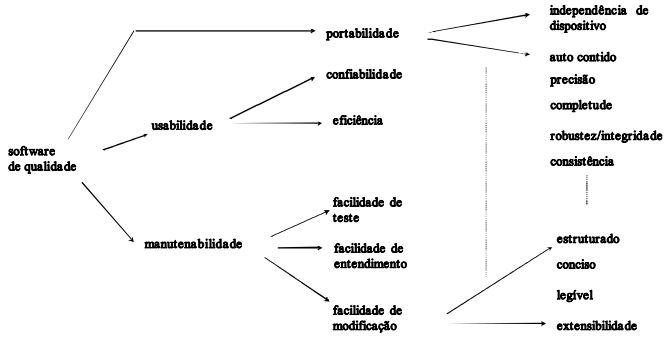
Métricas de Qualidade: motivação

- Várias métricas foram desenvolvidas para medir os atributos ou fatores de qualidade;
- Independentemente da métrica usada, sempre se busca os mesmos objetivos
  - Melhorar o entendimento da qualidade do produto;
  - Atestar a efetividade do processo;
  - Melhorar a qualidade do trabalho realizado a nível de projeto.

Métricas de Qualidade: exemplos

- Árvore de atributos de qualidade (Boehm, Brown e Lipow);
- Código fonte (Halstead);
- Qualidade da especificação (Davis);
- Métricas para sistemas orientados a objetos (OO).

Árvore de Atributos de Qualidade: Boehm, Brown e Lipow



Em matemática, um operando é uma das entradas (argumentos) de um operador. Por exemplo:

INPUT				OUTPUT
3	+	6	=	9
operando	operador	operando		

Código Fonte (Halstead)

- Baseia-se na habilidade de se obter as seguintes medidas primitivas num programa fonte ou estimadas na fase de projeto:
  - n1: numero de operadores distintos que aparecem num programa;
  - n2: número de operandos distintos que aparecem num programa;
  - N1: número total de ocorrência de operadores;
  - N2: número total de ocorrência de operandos.

Código Fonte (Halstead)

- Com base nestas medidas primitivas, Halstead definiu fórmulas para calcular outras métricas, como:
  - Comprimento global do programa;
  - Nível de programa: compara implementações de um algoritmo em linguagens diferentes;
  - Esforço de programação;
  - etc.
- Com estas medidas pode-se estimar o tempo total de programação, o número de erros esperados no programa, etc.

Qualidade da Especificação (Davis)

- Davis sugere uma lista de características que podem ser usadas para avaliar a qualidade do modelo e da correspondente especificação de requisitos:
  - Falta de ambiguidade;
  - Completude;
  - Corretude;
  - Facilidade de entendimento;
  - Verificabilidade;
  - Concisão;
  - Facilidade de rastreamento e de modificação, etc.

Qualidade da Especificação (Davis)
exemplo:

- Embora muitas medidas pareçam qualitativas, ele demonstrou que cada uma delas pode ser representada na forma de métricas;
- Exemplo:
  - **Falta de ambiguidade**:
    - Número de requisitos (nr = nf + nnf);
    - Número de requisitos para os quais os revisores de cada requisito tiveram a mesma interpretação (nui);
    - Idealmente a falta de ambiguidade (Q1 = nui/nr) = 1.

Qualidade da Especificação (Davis)
exemplo:

- Exemplo2:
  - **Falta de Completude**:
    - A completude dos requisitos funcionais pode ser computada por :
      - Q2=nu/(ni * ns)
        - » Onde:
        - » nu = Número de Requisitos funcionais únicos;
        - » ni = Número de Entradas definidas ou implicadas pela especificação;
        - » ns = Número de estados especificados;
      - Q2 mede a percentagem de funções necessárias que tenham sido especificadas para um sistema.

Qualidade da Especificação (Davis)
exemplo:

- Exemplo2:
  - **Falta de Completude**:
    - A completude dos requisitos **não-funcionais** deve considerar o grau de validação dos requisitos, pode ser computada por :
      - Q3=nc/(nc + nnv)
        - » Onde:
        - » nc = Número de Requisitos que foram validados;
        - » nnv = Número de Requisitos que ainda não foram validados;

Métricas para Sistemas Orientados a Objetos

- Estas métricas devem focalizar nas características que distingüem software OO de software convencional;
- Podem ser divididas em:
  - Métricas orientadas a classes;
  - Métricas para testes em OO;
  - Métricas para projeto OO.

Métricas Orientadas a Classes

- Métricas CK (Chidamber e Kemerer)
  - **Acoplamento entre objetos**: indica o grau de interdependência entre objetos. Quanto maior o acoplamento, menor é a reusabilidade da classe e mais difícil é a manutenção e os testes;
  - **Profundidade da árvore de herança**: quanto maior este valor, mais difícil é determinar o comportamento das classes de níveis mais baixos;
  - **Número de filhos**: com o crescimento do número de filhos, aumenta o reuso, mas as abstrações da superclasse podem ser diluídas;
  - etc.

Métricas Orientadas a Classes

- Métricas propostas por Lorenz e Kidd
  - **Tamanho da classe**: número total de operações mais o número de atributos. Quanto maior este número, maior a responsabilidade da classe, podendo reduzir a sua reusabilidade e dificultar a implementação e os testes;
  - **Número de operações redefinidas** por uma subclasse. Valores grandes para esta métrica geralmente indicam problemas de projeto (ex: violação da abstração);
  - **Número de operações adicionadas** a uma classe. Quanto maior este valor, mais específica é a classe e mais difícil é o seu reuso;
  - etc.

Métricas para Testes em OO

- Fornecem uma indicação da qualidade do projeto do esforço de testes requerido;
- São dividas em dois grupos relativos a características importantes do projeto:
  - Encapsulamento;
  - Herança.

Métricas para Testes em OO: encapsulamento

- **Coesão entre métodos**: quantidade de métodos de uma classe que acessam um mesmo atributo da classe. Se este valor é alto implica que mais estados devem ser testados para garantir que métodos não geram efeitos colaterais;
- **Percentagem de atributos públicos**: quanto maior este valor, mais testes precisam ser feitos na classe para garantir a ausência de efeitos colaterais;
- **Número de classes ou métodos que acessam atributos de outras classes**: quanto maior este valor, maior a possibilidade de violação de encapsulamento e maior a necessidade de testes.

Métricas para Testes em OO: herança

- **Número de classes raizes**: esta métrica indica o número de hierarquias distintas. Quanto maior o seu valor, maior é o esforço de testes;
- **Fan in**: indica a quantidade de classes herdadas por outra classe, ou seja, herança múltipla. Quanto maior o **fan in**, maior o esforço de teste;
- **Número de filhos** e **profundidade da árvore de herança**: quanto maior estes valores, maior a quantidade de métodos da superclasse que precisam ser retestados nas subclasses.

Métricas para Projeto OO

- O trabalho de um gerente é planejar, coordenar, verificar a evolução e controlar o projeto de software;
- A duração e o esforço requerido em um projeto são diretamente proporcionais ao tamanho do projeto;
- Métricas como as seguintes fornecem uma idéia do tamanho do software:
  - **Número de classes chaves**: classes que enfocam o domínio específico do negócio. Dificilmente podem ser implementadas exclusivamente via reuso. Quanto maior este valor, maior o esforço de desenvolvimento;
  - **Número de subsistemas**.

Controle e Garantia de Qualidade

- Definição:
  - "Atividade e técnica operacional que é utilizada para satisfazer os requisitos de qualidade" (McDermid).
- São funções gerenciais e estão relacionadas às atividades de verificação e validação.

Controle e Garantia de Qualidade

- Consome tempo no desenvolvimento de sistemas de software e vai além da entrega do sistema (entra na fase de manutenção);
- Técnicas usadas para cada atividade podem contribuir para o respectivo controle de qualidade;
- Algumas técnicas têm controle embutido, outras não.

65

Controle e Garantia de Qualidade

- Gerentes querem os melhores projetistas para projetar o produto, mas em geral não podem tê-los;
- Existe então a necessidade de concentrar esforços em métodos de SQA (Software Quality Assurance);
- O papel de SQA é monitorar os métodos e padrões que os engenheiros de software usam;
- Pessoas podem ser experientes em SQA sem, no entanto, serem experientes em projetos de software.

66

Atividades de SQA

- Em SQA temos uma variedade de tarefas, as quais podemos dividir em dois grandes grupos:
  - Engenheiros de software: fazem o desenvolvimento dos sistemas (trabalho técnico);
  - Grupo de SQA: responsabilidades sobre o plano de qualidade, inspeção, conservação de registros históricos, análise do produto desenvolvido e *reporting* das atividades de SQA ao gerente do projeto.

67

Atividades de SQA

- O SEI (Software Engineering Institute) recomenda as seguintes atividades para o grupo de SQA
  - Preparar uma plano de SQA;
  - Participar da descrição do projeto de software;
  - Revisar as atividades dos engenheiros de software;
  - Documentar e consertar os desvios;
  - Registrar discordâncias e reportar para o gerente;
  - Gerenciar mudanças e métricas de software.

68

Atividades de SQA: revisões de software

- São um filtro no processo de *ES*;
- Não são limitadas à especificação, projeto e código.
- Defeito ➡ anomalia do produto (IEEE);
- **R**evisões **T**écnicas **F**ormais (*RTF*) ➡ encontrar erros durante o processo antes que eles se tornem defeitos;
- 50% a 60% do total de erros são introduzidos durante o projeto de software;
- *RTF* podem descobrir cerca de 75% desses erros.

Atividades de SQA: medidas de produtividade de programação

- A qualidade do software depende da produtividade de programação, a qual é afetada por:
  - qualidade da documentação;
  - linguagem de programação;
  - disponibilidade de ferramentas;
  - experiência do programador;
  - comunicação no projeto;
  - grau de dependência entre módulos;
  - práticas de programação bem definidas.

Atividades de SQA: medidas de confiabilidade

- "Probabilidade de uma operação de programa de computador ser livre de **falha**".

não conformidade com os requisitos de software

- É um elemento importante para a qualidade do software;
- Exemplo: um software que opera corretamente em 96 das suas 100 execuções, tem uma confiabilidade de 0.96.

Confiabilidade x Segurança

- Confiabilidade
  - usa a análise estatística para determinar a probabilidade de que uma falha **venha a ocorrer.**
- Segurança
  - examina as maneiras segundo as quais as falhas resultam em condições que podem levar a uma deformação.

Plano de SQA

- Especifica os objetivos, as tarefas de *SQA* a serem realizadas, os padrões, os procedimentos a estrutura organizacional e os mecanismos de auditoria;
- Documentos de ES exigidos: Especificação de Requisitos, Descrição de Projeto, Plano (e Relatório) de Verificação e Validação, Documentação do Usuário.

Certificação de Qualidade

- Não basta que a qualidade exista, ela deve ser reconhecida pelo cliente;
- Deve existir uma certificação oficial emitida com base em um padrão;
- As certificações são dadas por instituições competentes;
- Exemplos de certificação:
  - Selo SIF de qualidade de produtos alimentícios;
  - Selo ABIC de qualidade do café;
  - Classificação da rede hoteleira (ex: hotél 5 estrelas).

Qualidade do Produto x Qualidade do Processo

- Hoje em dia, a qualidade do processo é mais importante do que a qualidade final do produto;
- Existe normas e padrões tanto para produtos quanto para processos.

Evolução dos Conceitos de Qualidade

Inspeção pós-produção	Produto final é avaliado depois de pronto.	1900
Controle estatístico de produção	Avalia subprodutos das etapas de produção	1940
Procedimento de produção	Avalia todo o procedimento de produção	1950
Educação das pessoas	Avalia as pessoas envolvidas no processo	1960
Otimização dos processos	Avalia e otimiza cada processo	1970
Projeto robusto	Avalia o projeto de produção	1980
Engenharia simultânea	Avalia a concepção do produto	1990

Padrões de Qualidade de Software

- Qualidade de produtos de software - ISO 9126 (versão brasileira - NBR 13596);
- Qualidade de pacotes de software - ISO 12119;
- Qualidade do processo de software
  - Capability Maturity Model (CMM)
  - Personal Software Process (PSP)
  - ISO 9000 / ISO 9001

77

Qualidade de produtos de software - ISO 9126

- Conjunto de características que devem estar presentes em um software de qualidade:
  - Funcionalidade - satisfaz as necessidades?
  - Confiabilidade - é imune a falhas?
  - Usabilidade - é fácil de usar?
  - Eficiência - é rápido e "enxuto"?
  - Manutenibilidade - é fácil de modificar?
  - Portabilidade - é fácil de usar em outro ambiente?
- Muitas destas características são subjetivas;
- Outras podem ser definidas por meio de métricas.

78

Qualidade de pacotes de software - ISO 12119

- Trata da avaliação de "software de prateleira";
- Descreve detalhes que devem estar presentes no software, 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 os dados de entrada;
  - etc.

79

Qualidade do processo de software

- Os estudos sobre qualidade estão na sua maioria voltados para o melhoramento do processo de desenvolvimento;
- Ao melhorar a qualidade do processo está se dando um grande passo para se garantir também a qualidade do produto.

80

Qualidade do processo de software - A Série ISO 9000

- "**Padrões de Gerenciamento e de Garantia de Qualidade - Diretrizes para Seleção e Uso**".
- Série de padrões *ISO 9000* ⇨ conjunto de documentos que trabalham com sistemas de qualidade que podem ser usados para propostas de garantia de qualidade externa.
- O *ISO 9000* descreve os elementos de sistemas de garantia de qualidade (estrutura organizacional, procedimentos, processos e recursos) em termos gerais.

ISO 9001

- "**Sistemas de Qualidade - Modelo para Garantia de Qualidade em Projeto, Desenvolvimento, Produção, Instalação e Serviço**";
- Aplicado para todas as engenharias.

ISO 9001: requisitos

- Define requisitos que devem estar presentes em um sistema de garantia de qualidade efetivo:
  - Gerência de responsabilidades
  - Sistema de qualidade documentado
  - Revisões de contrato
  - Controle de projeto
  - Controle do processo
  - Inspeções e testes
  - Inspeções, medidas e testes de equipamentos
  - Treinamento
  - *Servicing*
  - Técnicas estatísticas para verificar a aceitação do produto
  - etc.

ISO 9000-3

- Orientação para a aplicação da ISO 9001 no processo de Engenharia de Software;
- Todas as orientações giram em torno de uma "situação contratual", onde uma empresa contrata outra empresa para desenvolver um produto de software.

ISO 9000-3: processos definidos

- Estrutura do sistema de qualidade
  - Responsabilidade do fornecedor;
  - Responsabilidade do comprador;
  - Análise crítica conjunta.
- Atividades do ciclo de vida
  - Análise crítica do contrato;
  - Especificação de requisitos do comprador;
  - etc.
- Atividades de apoio
  - Gerenciamento de configuração;
  - Controle de documentos;
  - etc.

85

ISO 9000-3: processo de certificação

- 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;
- O órgão certificador 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.

86

Qualidade do processo de software - Capability Maturity Model (CMM)

- Descreve princípios e práticas relacionadas à maturidade do processo de software;
- Tem o objetivo de ajudar as organizações a melhorarem seus processos de software em termos de um caminho evolutivo que vai de *ad hoc* (processos caóticos) a processos maduros e disciplinados;
- Para isto define o conceito de **nível de maturidade**: base evolucionária bem definida direcionada a obter um processo de software maduro.

87

Capability Maturity Model (CMM): os 5 níveis de maturidade

Otimizado	Um processo de melhora contínuo é capacitado p/retorno quantitativo do processo e das idéias.
Gerenciado	Medidas de qualidade são coletadas. O processo e o produto são entendidos e controlados quantitativamente.
Definido	Processos padronizados, documentados e integrados.
Reproduzível	Processos estabelecidos por experiências anteriores.
Inicial	Processo caótico e ad hoc.

88

Capability Maturity Model (CMM): áreas-chave de processo (KPA)

- Indicam as áreas que uma organização deveria enfocar para melhorar seu processo de software;
- O CMM define 18 KPA's distribuídas nos 5 níveis;
- Cada *KPA* é descrita em termos de **práticas** que contribuem para satisfazer seus objetivos.
  - descrevem a infra-estrutura e atividades que contribuem para a implementação e institucionalização da *KPA*.

Capability Maturity Model (CMM): áreas-chave de processo (KPA)

- **KPA 1**: não existem KPA's para este nível;
- **KPA 2**: interesses relacionados ao estabelecimento do controle básico de administração de projeto;
- **KPA 3**: problemas organizacionais e de projeto;
- **KPA 4**: estabelecer um entendimento quantitativo do processo de software e do produto;
- **KPA 5**: cobrem os problemas que a organização e os projetos devem endereçar para implementar uma melhora contínua e mensurável do processo de software.

Comparação entre ISO 9001 e CMM

- CMM
  - Ênfase no contínuo processo de melhora;
  - Enfoca estritamente o software;
  - Não é uma norma emitida por uma instituição de padronização.
- ISO 9001
  - Ênfase no critério mínimo para um sistema de qualidade aceitável;
  - Tem um escopo mais abrangente;
  - Por ser mais conhecido e embutir um padrão internacional mínimo de qualidade, o ISO talvez traga melhores resultados para a empresa.

Comparação entre ISO 9001 e CMM - perguntas

- Em que nível do *CMM* poderia se encaixar uma organização em conformidade com o *ISO 9001*?
- Uma organização de nível 2 (ou 3) poderia ser considerada em conformidade com o *ISO 9001*?
- Meus esforços na melhoria do processo e no gerenciamento de qualidade deveriam ser baseados no *ISO 9001* ou no *CMM*?
- Estas perguntas não têm uma resposta definitiva.

Qualidade do processo de software - Personal Software Process (PSP)

- O estímulo para desenvolver o *PSP* surgiu de questões sobre o *CMM*;
- Estratégia para o desenvolvimento pessoal;
- Objetivo: fazer com que os engenheiros de software fiquem atentos ao processo que eles usam e estejam sempre verificando suas performances no processo de desenvolvimento, aumentando assim a produtividade.

Personal Software Process (PSP)

- Usando o PSP, os engenheiros de software:
  - desenvolvem um plano para todo projeto;
  - registram seu tempo de desenvolvimento;
  - trilham seus defeitos;
  - mantêm dados de um projeto em relatórios resumidos;
  - usam esses dados para planos de projetos futuros;
  - analisam dados que envolvem seus processos a fim de aumentar suas performances.

Conclusão

- Qualidade é um conceito complexo, porque significa diferentes coisas para diferentes pessoas;
- Não há uma simples medida para qualidade de software que seja aceitável para todos os projetos de todas as empresas;
- Para estabelecer ou melhorar a qualidade de software, deve-se definir os aspectos de qualidade nos quais se está interessado e, então, decidir como fazer para medí-los;
- Apesar dos custos elevados, é importante introduzir sistemas de gerenciamento de qualidade de software, como o *CMM* ou o *ISO 9001*.