

Qualidade de Software

Visão Geral

Two horizontal bars are positioned below the title. The top bar is a solid blue line, and the bottom bar is a grey line with a blue gradient, appearing as a shadow or a secondary bar.

Simone Senger Souza
srocio@icmc.usp.br

ICMC/USP

Qualidade de Software

- O que é qualidade?
- Como medir?



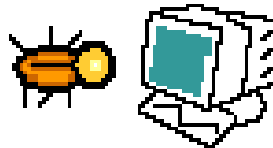
Visão de Qualidade de Software

- Defeito zero
- Grande número de funções
- Codificação elegante
- Alto desempenho
- Baixo custo de desenvolvimento
- Desenvolvimento rápido
- Facilidade para o usuário

Bugs históricos

- Piores bugs da história do software:

- <http://www.wired.com/software/coolapps/news/2005/11/69355?currentPage=all>



Bugs históricos

■ Mariner I – 1962

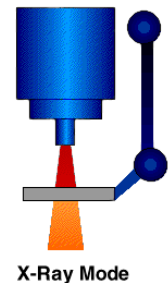
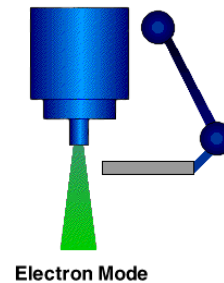
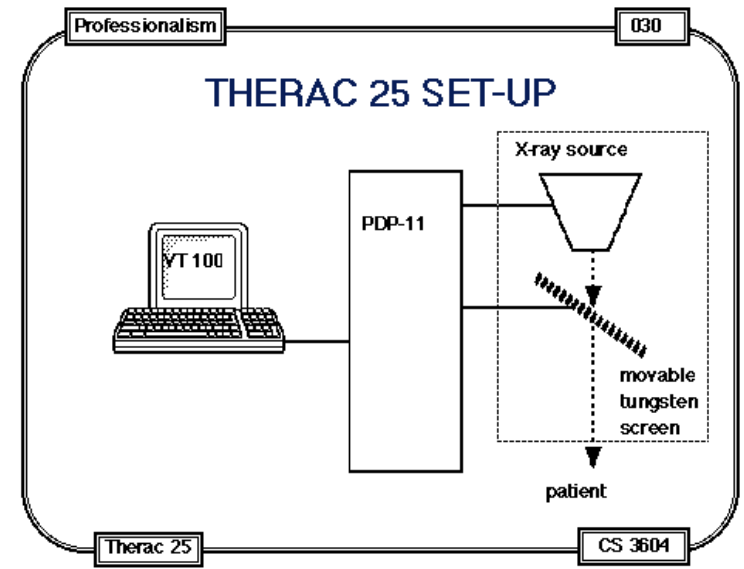
- Missão observar planeta Vênus
- Fórmula matemática foi equivocadamente transcrita para o computador
- Desviou do curso e foi destruído 4 min após lançamento
- Prejuízo: US\$ 18,5 mi



Bugs históricos

■ Therac-25 – 1985/1987

- Dispositivo de terapia por radiação sobre células cancerosas
- Libera doses letais de radiação em vários consultórios médicos
- Condição de disputa no SO
- 5 mortes, várias pessoas feridas



Bugs históricos

■ **Míssil Patriot – 1991**

- Míssil de defesa
- Dhahran, Arábia Saudita
- Erro de software no relógio do míssil: a cada 100 horas o relógio interno do sistema desviava um terço de segundo
- Recomendação era reiniciar o sistema de tempos em tempos
- Resultado: 600 metros de erro na distância em uma interceptação
- 28 soldados americanos mortos



Bugs históricos

■ Divisão de pontos flutuantes nos processadores Pentium da Intel – 1993

- Erro em divisões dentro de uma faixa de números (erro ~0,006% no arredondamento)
- 3 a 5 milhões de peças com defeito
- *Recall* para todos que quiseram trocar
- Custou à Intel US\$ 475 milhões



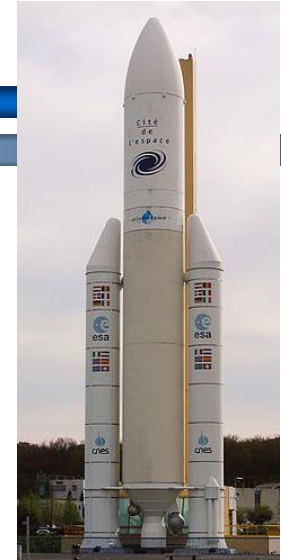
$$\frac{4195835}{3145727} = 1.333820449136241002$$

$$\frac{4195835}{3145727} = 1.333739068902037589$$

Bugs históricos

■ Ariane 5 voo 501 – 1996

- Levou uma década de desenvolvimento e custou 7 bilhões de dólares.
- Foguete com código reutilizado do Ariane 4 (outro hardware);
- *Overflow* de inteiro: conversão de *float* de 64-bits para inteiro 16-bits com sinal;
- O processador primário do foguete sobrecarrega os motores que se desintegraram em 40 segundos;
- Não tripulado (sem vítimas); prejuízo de US\$ 370 milhões



Bugs históricos

■ Bug do milênio (Y2K) – 2000

- Datas com apenas 2 dígitos para o ano
- Uma das maiores histerias da história
- Ao virar o ano 2000, a preocupação era que contasse como 1900
- Entre US\$ 300 e US\$ 500 bi no mundo todo



Bugs históricos

■ Toyota Prius – 2010

- Problema no software do sistema ABS de freios – acelerador fica preso, dificultando desaceleração
- Recall de 400.000 veículos
- ~ US\$ 2 bilhões de prejuízo, desvalorização de 15% nas ações



Bugs históricos

■ Play Station Network - 2011

- Invasão do sistema
- Dados privados e de cartão de crédito de ~70 mi de pessoas foram roubados



PLAYSTATION®
Network



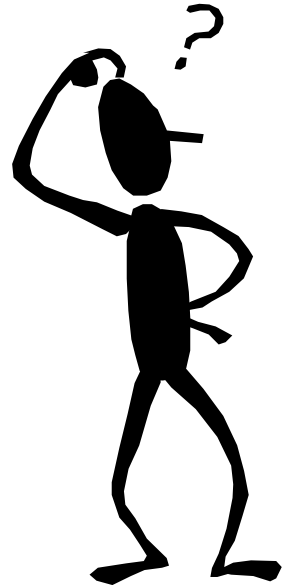
26/04/2011 18h03 - Atualizado em 26/04/2011 19h59

Dados pessoais de usuários da PSN foram roubados, admite Sony

Empresa não descarta roubo de informações de cartões de crédito.
PlayStation Network está fora do ar há seis dias.

Bugs históricos

- Por que essas falhas ocorrem?
- Poderiam ser evitadas?
- De quem é a culpa?



Atividades do Processo de Desenvolvimento



Atividades Guarda-Chuva

- Controle de Projeto
- Revisões Técnicas Formais
- **Garantia de Qualidade**
- Gerenciamento de Configuração
- Gestão de Reutilização
- Medição
- Gestão de Risco

Uma Visão Genérica: 3 Fases

1. Definição - “o que”

- Engenharia do Sistema
- Planejamento do Projeto
- Engenharia de Requisitos

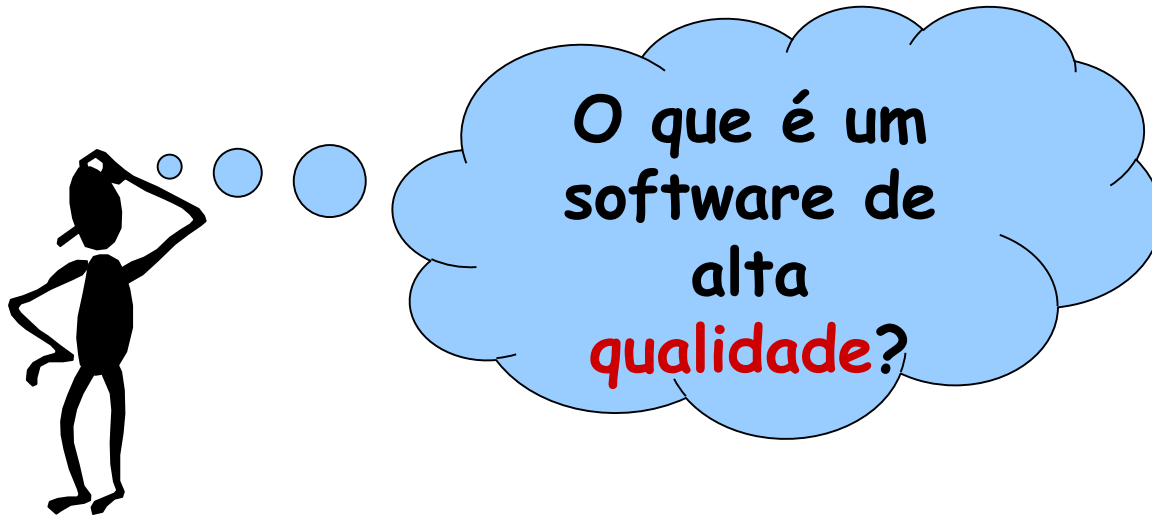
2. Desenvolvimento - “como”

- Projeto
- Geração do Código
- Teste

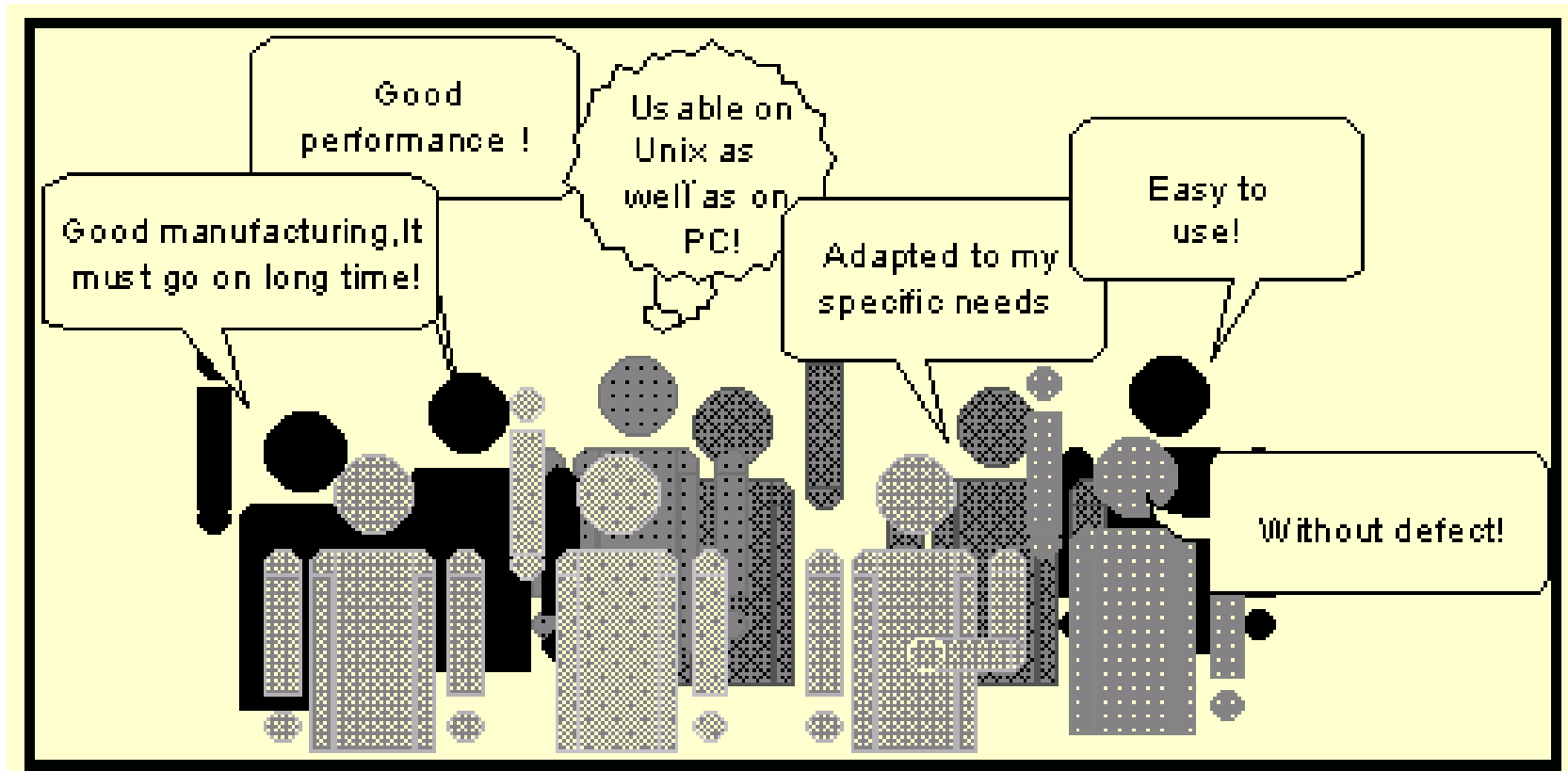
3. Manutenção

Engenharia de Software

- A **Engenharia de Software** é uma disciplina que aplica os princípios de engenharia com o objetivo de produzir software de **alta qualidade** a **baixo custo**.



Qualidade de Software



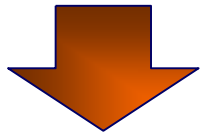
A Qualidade depende do Tipo de Aplicação

Sistema de Missão
Crítica

Software Embarcado

EXEMPLO

Qualidade Importante



Fazer aquilo que eu quero



Comportar-se com precisão



Ser fácil de usar



Rodar bem no hardware



Fácil de alterar



A Qualidade depende do Tipo de Aplicação

Software para Folha de Pagamento

Software Interativo com o usuário

EXEMPLO

Qualidade Importante



Fazer aquilo que eu quero



Se comportar com precisão



Ser fácil de usar



Rodar bem no hardware



Fácil de alterar



Qualidade de Software

“A qualidade de um projeto engloba o grau de **atendimento às funções e características especificadas** no modelo de requisitos”

[Pressman,2011]

satisfação do usuário = produto compatível + boa
qualidade + entrega no prazo + entrega dentro do
orçamento

Aspectos Importantes da Definição de Qualidade

1- Os requisitos de software são a base a partir da qual a qualidade é medida.

A falta de conformidade aos requisitos significa falta de qualidade.

Aspectos Importantes da Definição de Qualidade

2- Existe um conjunto de **requisitos implícitos** que freqüentemente não são mencionados na especificação. Por exemplo, o desejo de uma boa **manutenibilidade**.

Se o software atende aos requisitos explícitos, mas falha nos requisitos implícitos, a qualidade é suspeita.

Aspectos Importantes da Definição de Qualidade

3 - Existe, ainda, uma visão de qualidade de software do ponto de vista **gerencial**.

- O software é considerado de qualidade desde que possa ser desenvolvido dentro do prazo e do orçamento especificados.

A Qualidade depende do Ponto de Vista



usuário

O interesse fica concentrado principalmente no uso do software: facilidade de uso, requisitos atendidos.



desenvolvedor

A qualidade fica mais voltada às características internas do software: legibilidade, testabilidade, eficiência.

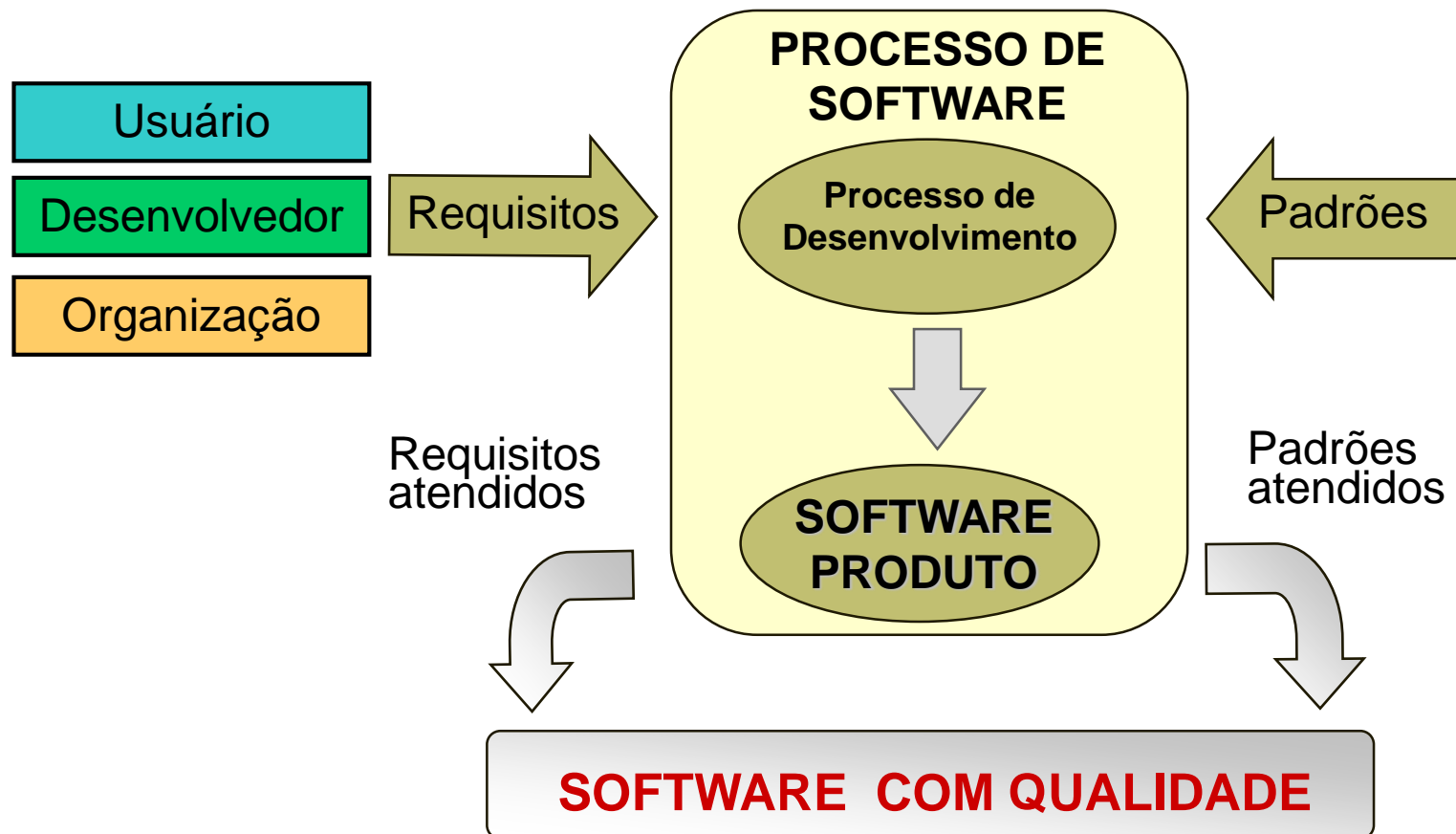


gerente

A qualidade do produto não pode ser desvinculada dos interesses da organização: custos e prazos.

Requisitos de Software

Base da Qualidade



Incorporação da Qualidade

Requisitos do Usuário

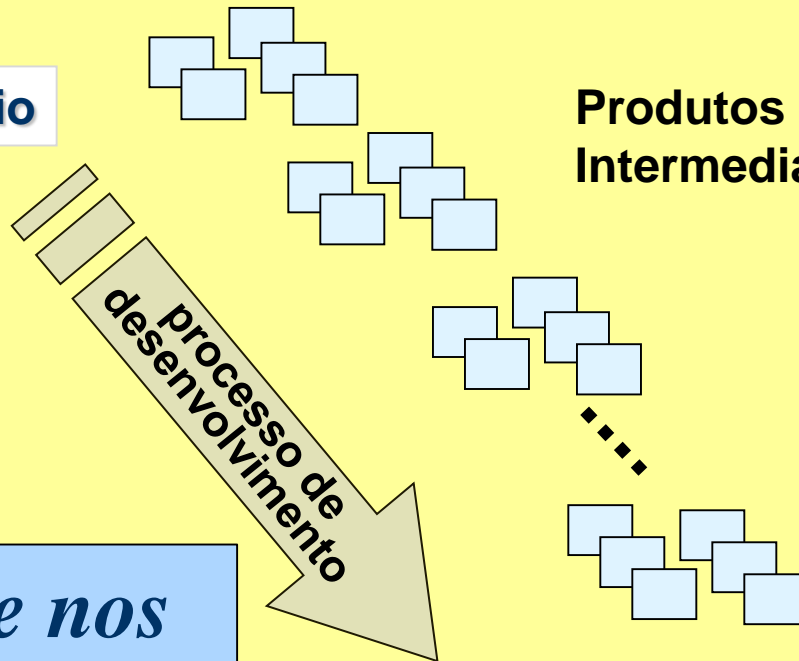
Produtos Intermediários

Produto Final

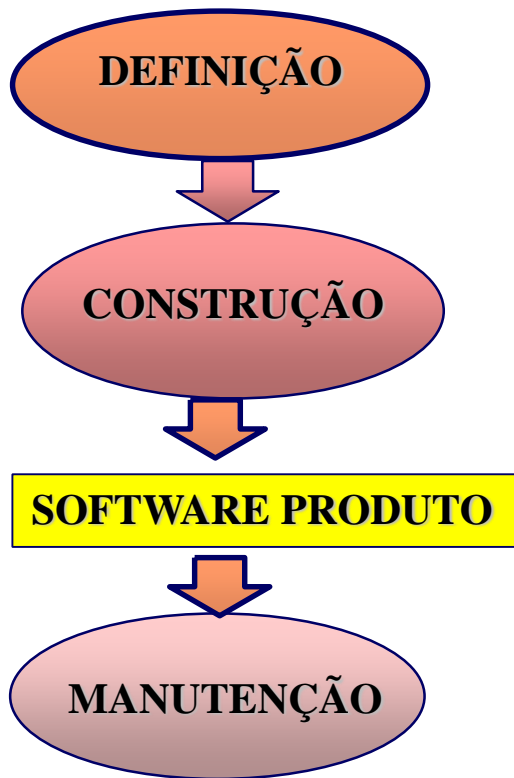
processo de desenvolvimento

Entrega do Produto Final

Qualidade presente nos produtos intermediários

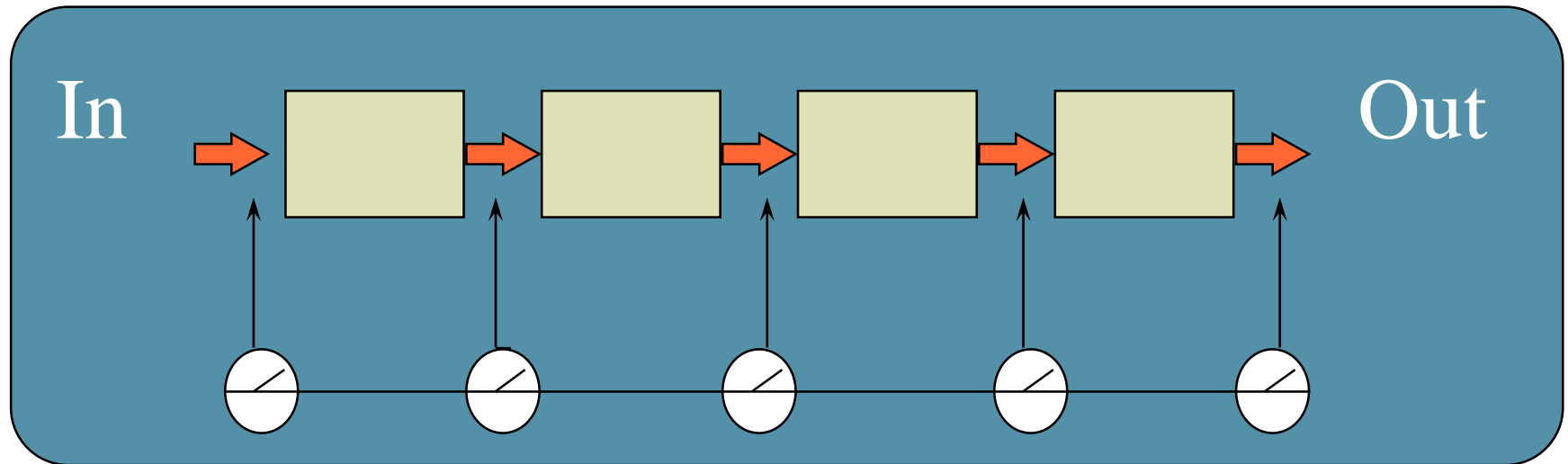


Qualidade de Software



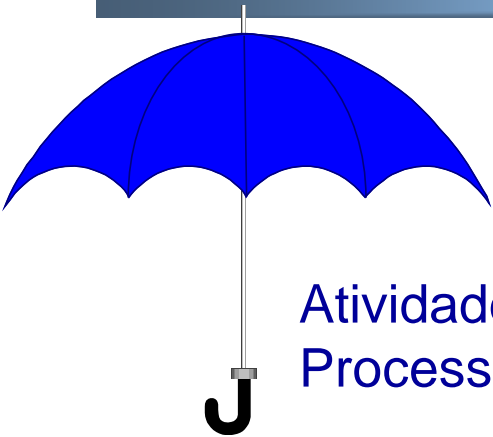
- A qualidade não pode ser incorporada ao produto depois de pronto.
- Para que a qualidade possa ser efetivamente incorporada ao **produto**, ela deve ser um objetivo constante do **processo de desenvolvimento**.

Garantia de Qualidade de Software



A **Garantia da Qualidade de Software (SQA)** é uma série planejada de **atividades de apoio** que atribui confiança ao software, as quais utilizadas em todo o processo de desenvolvimento

Garantia de Qualidade de Software



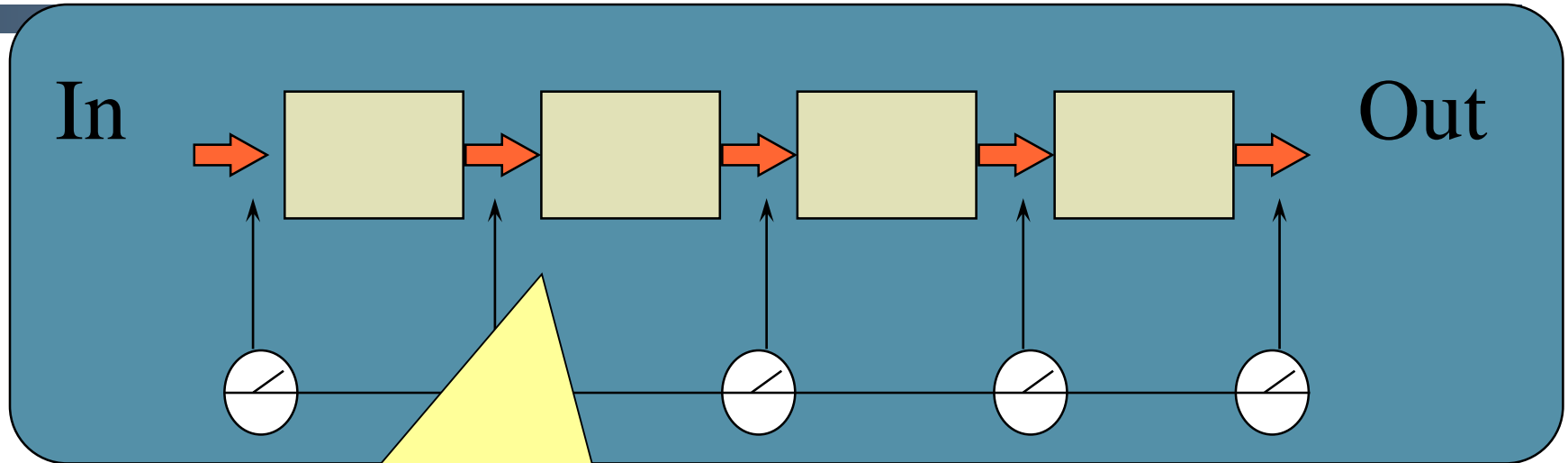
Atividades Guarda-Chuva do
Processo de desenvolvimento:

- Controle de Projeto
- Revisões Técnicas Formais
- **Garantia de Qualidade**
- Gerenciamento de Configuração
- Gestão de Reutilização
- Medição
- Gestão de Risco



Métodos técnicos
Revisões técnicas formais
Teste de software
Aplicação de padrões
Controle de mudanças
Medição
Manutenção de registros

Garantia de Qualidade



Como avaliar a qualidade dos
produtos ???

Garantia de Qualidade

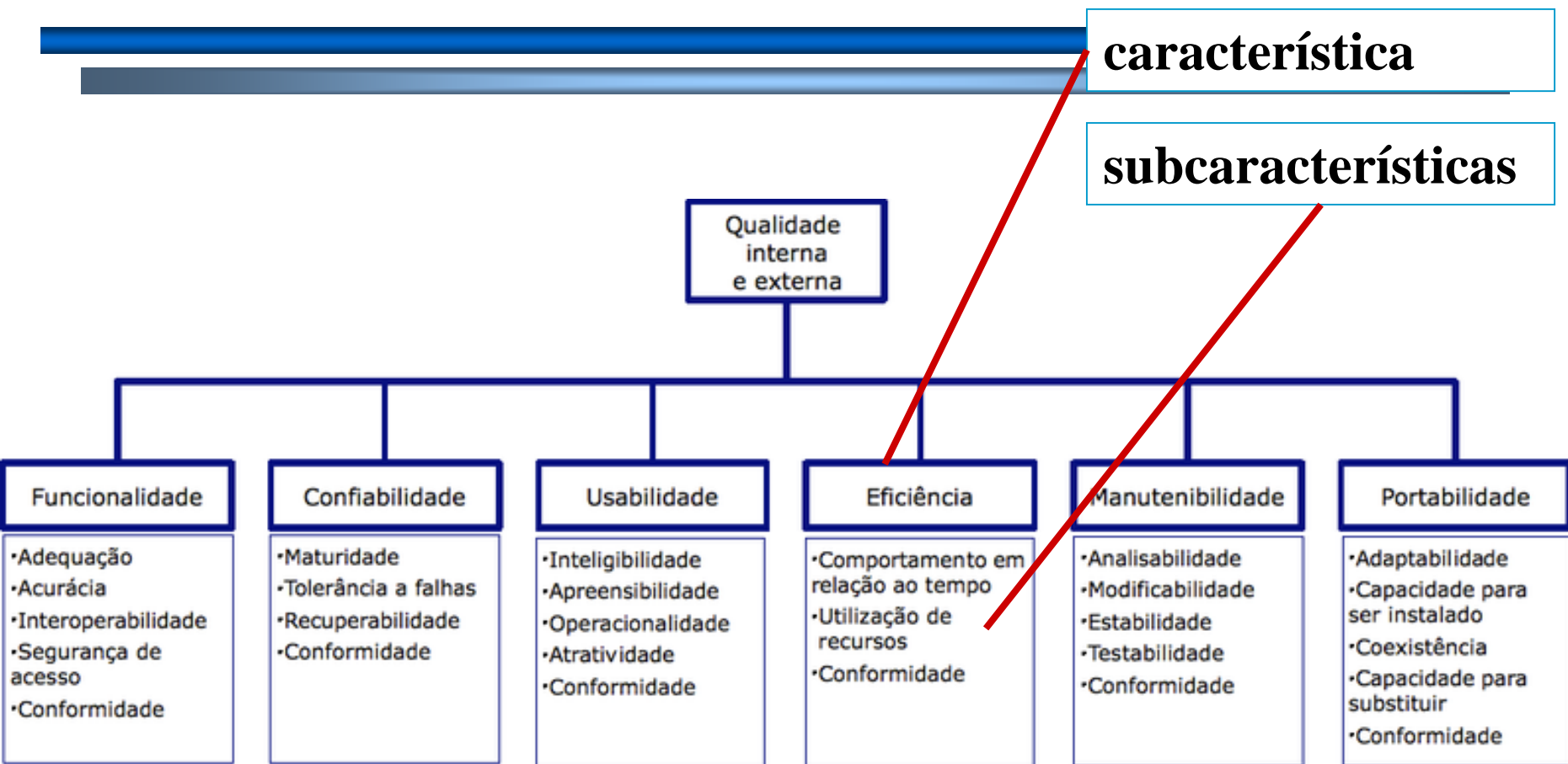
■ Algumas Normas

- [ISO 9126 – Qualidade de produto de software](#)
- ISO 12207 – Qualidade do processo de software
- ISO 27000 – Segurança da informação
- IEEE 829 – Documentação de testes
- IEEE 1028 – Revisão de software
- IEEE 1044 – Classificação de incidentes

Norma ISO/IEC 9126

- Baseada em três níveis:
 - Características, Sub-características e Métricas.
 - Cada **característica** é refinada em um conjunto de **sub-características** e cada sub-característica é avaliada por um conjunto de **métricas**.

Norma ISO/IEC 9126



Norma ISO/IEC 9126

FUNCIONALIDADE - Satisfaz as necessidades implícitas e explícitas do usuário?

SUBCARACTERÍSTICA

- **Adequação**
- **Acurácia**
- **Interoperabilidade**
- **Conformidade**
- **Segurança de Acesso**

PERGUNTA-CHAVE

- **É adequado as necessidades do usuário?**
- **Faz o que foi proposto de forma correta?**
- **É capaz de interagir com os sistemas especificados?**
- **Está de acordo com as normas, leis, etc. relacionadas à funcionalidade?**
- **Evita acesso não autorizado a programas e dados?**

Norma ISO/IEC 9126

CONFIABILIDADE - o software, durante um período de tempo, funciona de acordo com as condições pré-estabelecidas?

SUBCARACTERÍSTICA

- **Maturidade**
- **Tolerância a Falhas**
- **Recuperabilidade**
- **Conformidade**

PERGUNTA-CHAVE

- Com que frequência apresenta falhas?**
- Ocorrendo falhas, como ele reage?**
- É capaz de recuperar dados após uma falha?**
- Está de acordo com as padrões, normas, etc. relacionadas à confiabilidade?**

Norma ISO/IEC 9126

USABILIDADE – O software é fácil de usar?

SUBCARACTERÍSTICA

- Intelegibilidade

- Apreensibilidade

- Operacionalidade

- Atratividade

- Conformidade

PERGUNTA-CHAVE

É fácil entender os conceitos utilizados?

É fácil aprender a usar?

É fácil operar e controlar?

É atrativo ao usuário?

Está de acordo com as padrões, normas, etc. relacionadas à usabilidade?

Norma ISO/IEC 9126

EFICIÊNCIA – O software não desperdiça recursos?

SUBCARACTERÍSTICA

- Comportamento em Relação ao Tempo
- Comportamento em Relação aos Recursos
- Conformidade

PERGUNTA-CHAVE

- Qual é o tempo de resposta e de processamento?
- Quanto recurso usa? Durante quanto tempo?
- Está de acordo com as normas, leis, etc. relacionadas à eficiência?

Norma ISO/IEC 9126

MANUTENIBILIDADE – O software é fácil de alterar?

SUBCARACTERÍSTICA

PERGUNTA-CHAVE

- **Analísabilidade** **É fácil encontrar um defeito, quando ocorre?**
- **Modificabilidade** **É fácil modificar e remover defeitos?**
- **Estabilidade** **Existe risco de efeitos inesperados quando se faz alterações?**
- **Testabilidade** **É fácil testar o software modificado?**
- **Conformidade** **Está de acordo com as normas, leis, etc.? relacionadas à manutenibilidade?**

Norma ISO/IEC 9126

PORTABILIDADE - É fácil de usar em outro ambiente?

SUBCARACTERÍSTICA

PERGUNTA-CHAVE

- **Adaptabilidade** **É fácil adaptar a ambientes diferentes?**
- **Capacidade para ser instalado** **É fácil instalar?**
- **Capacidade para substituir** **É fácil usar para substituir outro?**
- **Conformidade** **Está de acordo com as normas, leis, etc. relacionadas à portabilidade?**
- **Co-existência** **Pode coexistir com outros produtos independentes compartilhando recursos?**

Métricas de Qualidade

- Como medir a qualidade de um software?
 - Funcionalidade?
 - Confiabilidade?
 - Usabilidade?
 - Eficiência?
 - Manutenibilidade?
 - Portabilidade?



Métricas de Qualidade

- Como medir a qualidade de um software?
 - Métricas dinâmicas
 - Durante teste ou uso do sistema
 - Durante depuração
 - Métricas estáticas
 - Informações coletadas do código



Métricas de Qualidade



■ Exemplos de métricas estáticas:

- Fan-in/fan-out
 - Fan-in: número de módulos que chamam o módulo x ;
 - Fan-out: número de módulos chamados por x .
- Tamanho do código (LoC)
- Quantidade de identificadores
- Complexidade ciclomática

Exemplo – Complexidade Ciclomática

```
void bolha(int a[], int size) {
/*1*/   int i, j, aux;
/*2*/   for (i = 0; i < size; i++)
/*3*/       for (j = size - 1; j > i; j--)
/*4*/           if (a[j - 1] > a[j])
/*5*/               {
/*6*/                   aux = a[j - 1];
/*7*/                   a[j - 1] = a[j];
/*8*/                   a[j] = aux;
/*9*/               }
/*10*/ }
```

A complexidade ciclomática define o número de **caminhos independentes** de um programa.

Exemplo – Complexidade Ciclomática

```
void bolha(int a[], int size) {
/*1*/   int i, j, aux;
/*2*/   for (i = 0; i < size; i++)
/*3*/       for (j = size - 1; j > i; j--)
/*4*/           if (a[j - 1] > a[j])
/*5*/               {
/*6*/                   aux = a[j - 1];
/*7*/                   a[j - 1] = a[j];
/*8*/                   a[j] = aux;
/*9*/               }
/*10*/ }
```

Complexidade ciclomática =

num de decisões binárias + 1

Exemplo – Complexidade Ciclomática

```
void bolha(int a[], int size) {  
/*1*/   int i, j, aux;  
/*2*/   for (i = 0; i < size; i++)  
/*3*/       for (j = size - 1; j > i; j--)  
/*4*/           if (a[j - 1] > a[j])  
/*5*/               {  
/*6*/                   aux = a[j - 1];  
/*7*/                   a[j - 1] = a[j];  
/*8*/                   a[j] = aux;  
/*9*/               }  
/*10*/ }  
}
```

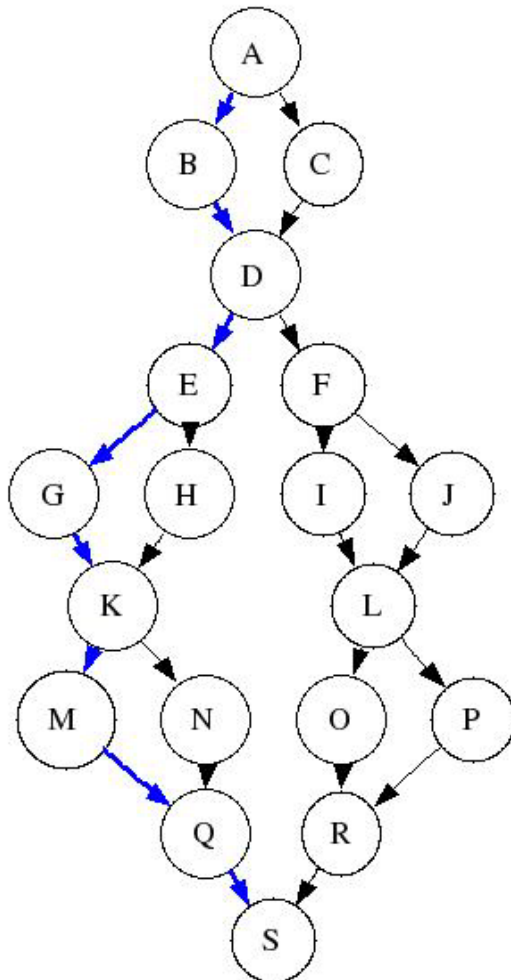
Complexidade ciclomática = 3 + 1

Métricas de Qualidade



- Complexidade ciclomática:
 - Auxílio no teste do software (confiabilidade e funcionalidade)
 - Auxílio na manutenibilidade:
 - De 0 a 5: Seu código está, provavelmente, ok;
 - Entre 6 e 10: Pense em maneiras de simplificar o código;
 - Maior que 10: Código precisa ser dividido

Métricas de Qualidade



Exemplo de grafo
de programa com
Complexidade
ciclomática = 7

Métricas de Qualidade



- A maioria das métricas são obtidas da validação/verificação do software
 - Teste de software
 - Revisão (Inspeção)
 - ...

Métrica de Qualidade



■ Exemplo: Checklist para Revisão de DR

ITEM	ITEM PARA VERIFICAÇÃO	SIM	NÃO
Não ambíguo: É não ambíguo se, e somente se, cada requisito declarado seja suscetível a apenas uma interpretação.			
1	Cada requisito está descrito com clareza, concisão e sem ambiguidade?		
Consistente: É consistente se, e somente se, nenhum dos requisitos do documento, tomado individualmente, está em conflito com qualquer outro requisito do mesmo documento.			
2	Existem requisitos conflitantes?		
Completo: É completo se, e somente se, conter toda e apenas a informação necessária para que o software correspondente seja produzido.			
3	Existem requisitos implícitos?		
4	Os requisitos exibem a distinção clara entre funções, dados e restrições?		
5	As restrições e dependências foram claramente descritas?		
6	Existem requisitos que contêm algum nível desnecessário de detalhe do projeto?		
7	Os requisitos definem todas as informações a serem apresentadas aos usuários?		
8	Os requisitos descrevem as respostas do sistema ao usuário devido às condições de erro?		
9	Existem situações não tratadas pelos requisitos que precisam ser consideradas?		
10	O documento contém realmente toda a informação prometida em sua introdução?		

Métrica de Qualidade



- Exemplo: Checklist para Revisão de Usabilidade de Software

