Revisão para a Prova 3

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

dcc603@dcc.ufmg.br

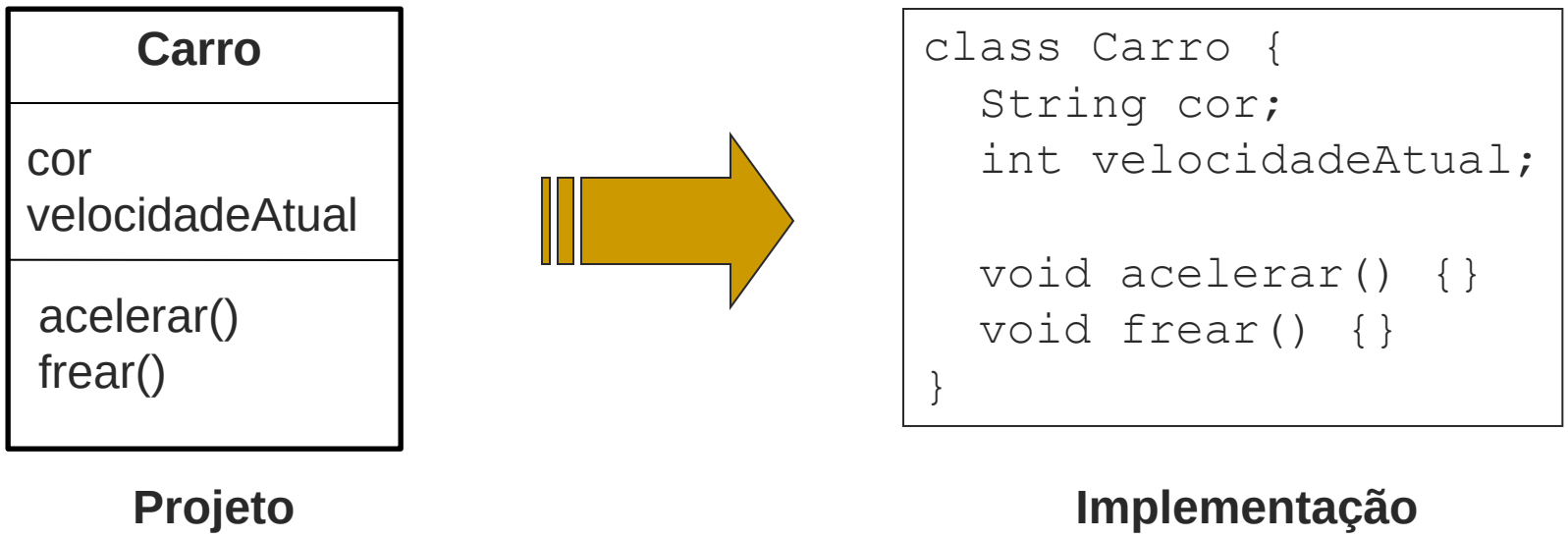
14 Outubro 2020

18 - POO, Java e Idiomas

- Um programa OO é geralmente constituído de várias classes
  - Cada classe possui vários métodos (comportamento)
  - Classes também possuem atributos (estados)
- Objetos são criados por uma classe
  - Objetos trocam mensagens pela chamada de métodos

Do Projeto para Implementação

- Antes do carro ser codificado, ele deve ser projetado



Associação em Java

```
public class AlarmClock {

   protected Time alarm;

   public void setAlarm(Time1 time) { alarm = time; }
   ...
}
```

Herança em Java

```
// Pessoa.java
public class Pessoa {
   protected String nome;
   private String email;
   public void enviarMensagem() { ... }
}

// Aluno.java
public class Aluno extends Pessoa {
   private String matricula;
}

// Professor.java
public class Professor extends Pessoa {
}
```

Membros de Uma Classe

- Construtor
- Métodos
  - De classe
  - De objeto
- Variáveis
  - De classe
  - De objeto
- Constantes

Idiomas

- Idiomas são padrões de baixo nível específicos de uma linguagem de programação
- Cada idioma descreve como resolver um problema de programação em uma determinada linguagem
- Idiomas facilitam a comunicação entre programadores
  - Aceleram o desenvolvimento
  - Facilitam atividades de manutenção

Bibliografia

- DEITEL, H. M.; DEITEL P. J. **Java: Como Programar**, 8a. Edição. Pearson, 2010.
  - Seções 1.5 a 1.10; Capítulos 3, 8 e 9
- F. Buschmann et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.
  - Cap. 4 Idioms
- A. von Staa. **Programação Modular**. Elsevier, 2000.
  - Apêndices 3, 4 e 5

19 - Exercício Java

- Implementar o sistema em Java
  - Seguir os idiomas de programação
  - Considerar os objetos principais do seu Diagrama de Classes

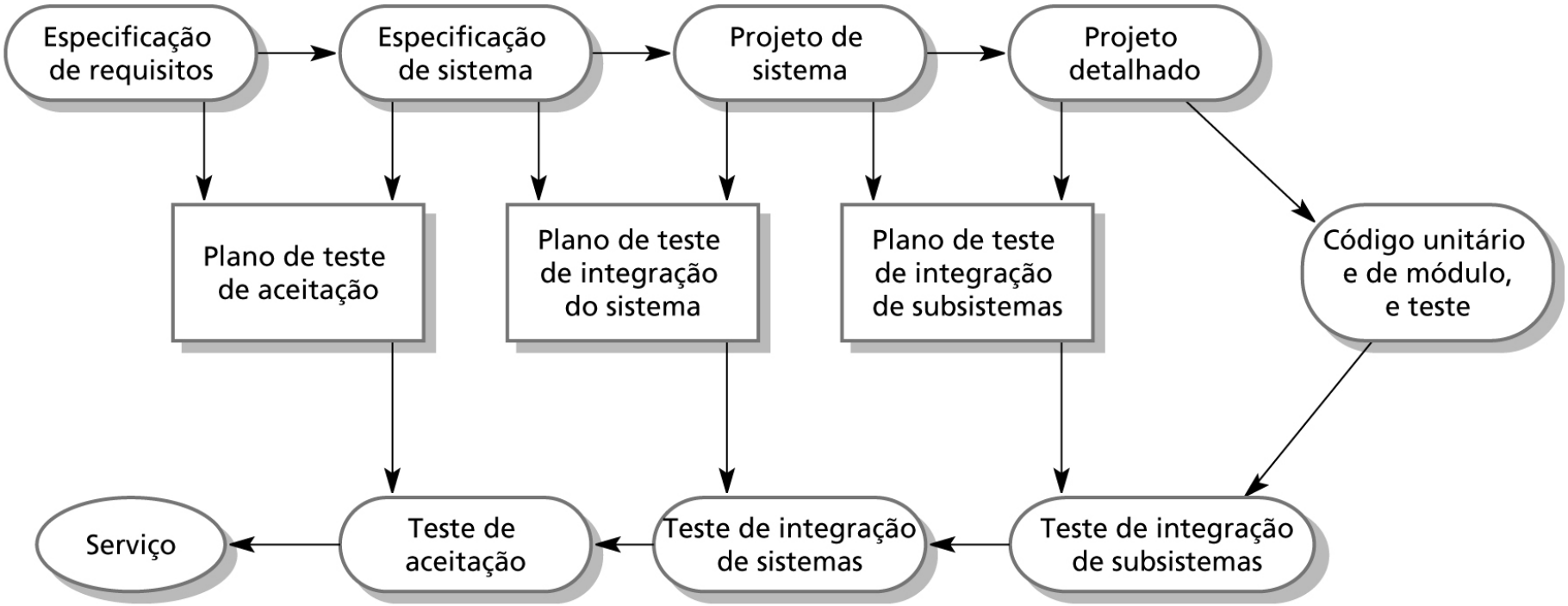
- Não precisa estar 100% implementado
  - Vou olhar principalmente os conceitos e idiomas de programação OO

20 - Verificação e Validação

- Objetivos da verificação e validação
  - Mostrar que o software atende a sua especificação
  - Mostrar que o software atende as necessidades do cliente

- Teste é a principal técnica de V&V
  - Técnicas de inspeção e revisão também são usadas

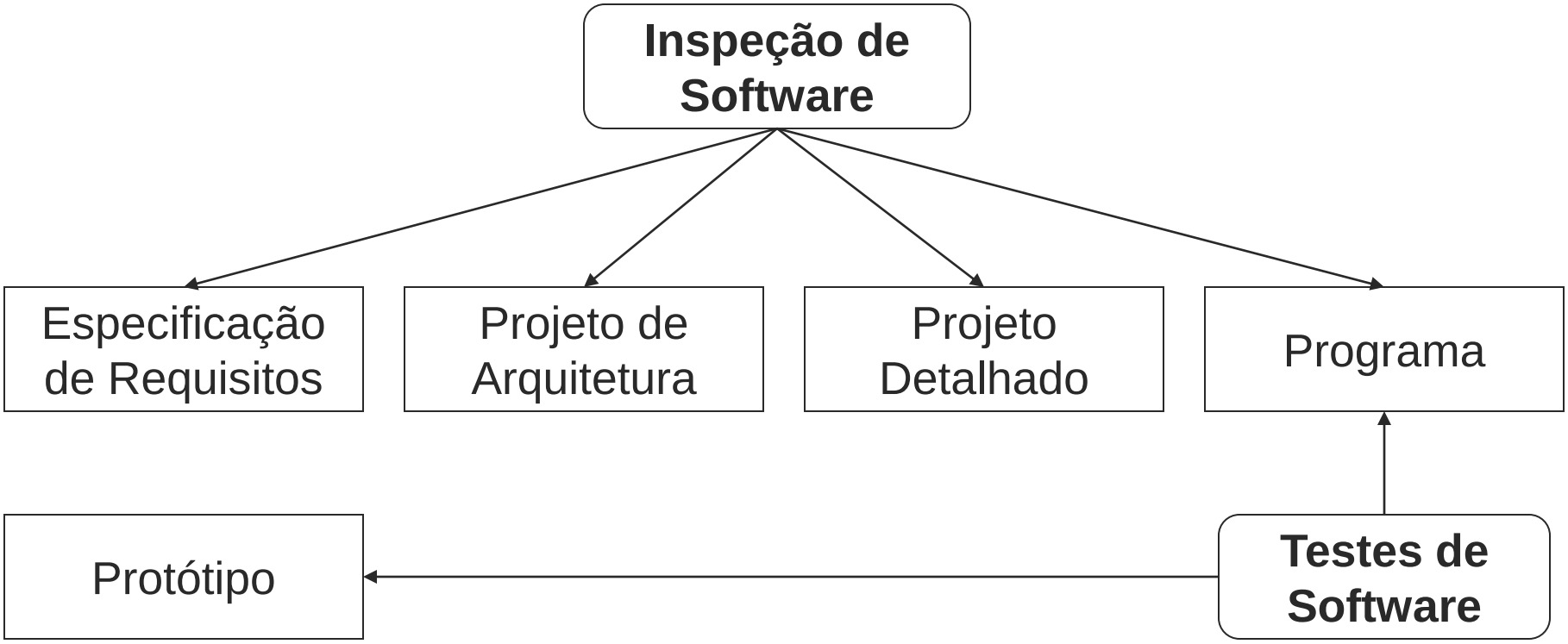
O Modelo V


Especificação de requisitos
Especificação de sistema
Projeto de sistema
Projeto detalhado
Plano de teste de aceitação
Plano de teste de integração do sistema
Plano de teste de integração de subsistemas
Código unitário e de módulo, e teste
Serviço
Teste de aceitação
Teste de integração de sistemas
Teste de integração de subsistemas

Inspeções de Software

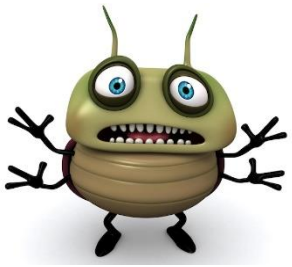
- Técnica estática de V&V
  - Não precisa executar o programa
- Pode ser usada em qualquer atividade de desenvolvimento
  - Requisitos, projeto, código fonte, etc.
- Pode ser semi-automatizada por análise estática
  - Análise não automatizada é cara

Verificação Estática e Dinâmica



Testes de Software

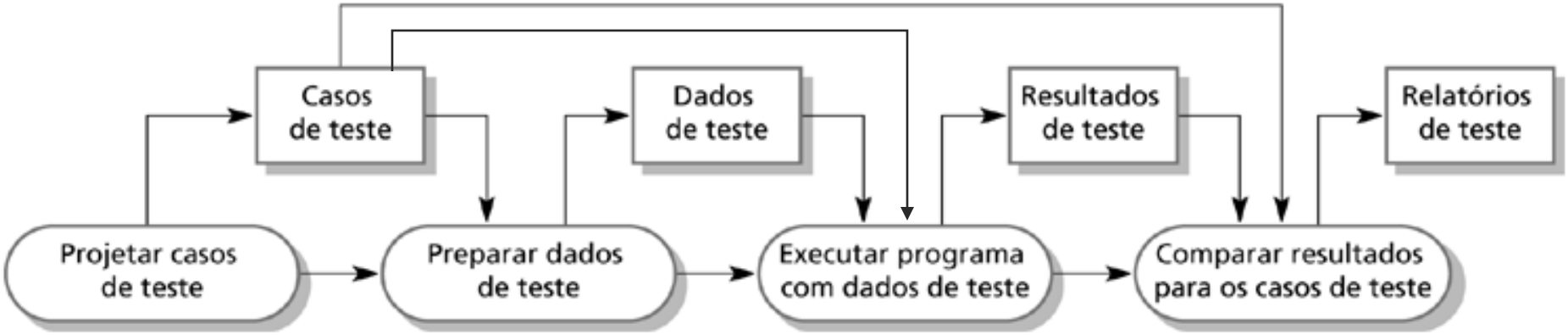
- Teste de software buscam por erros ou anomalias em requisitos funcionais e não funcionais



- Classificação de testes pelo objetivo
  - **Teste de Validação**: mostrar que um programa faz o que é proposto a fazer
  - **Teste de Defeito**: descobrir os defeitos do programa antes do uso

Processo de Teste

- Exemplo de processo de teste baseado em planos
  - As atividades são planejadas antes de serem executadas



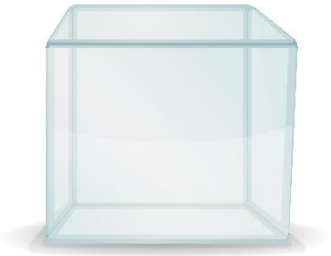
Teste de Unidade

- Objetivo é garantir que uma unidade ou classe funciona
  - Testa unidades individuais de programa de forma independente
- Geralmente é de responsabilidade do próprio desenvolvedor da unidade
  - Os testes são derivados da experiência do desenvolvedor

Teste de Partições

- Dados de entrada e resultados de saída podem ser particionados
  - O programa se comporta de maneira semelhante para cada partição
- Exemplos de partições
  - Números positivos / negativos
  - Itens de um mesmo menu
- Casos de teste devem ser escolhidos para exercitar cada partição

Teste Estrutural



- Frequentemente chamado de teste caixa-branca
- A escolha de casos de teste ocorre de acordo com a estrutura do programa
  - O conhecimento do programa é usado para identificar casos de teste
- O objetivo é exercitar todas as declarações do programa

Teste de Integração

- O objetivo é garantir que dois ou mais unidades funcionam juntos
  - Testa grupos de unidades integradas para criar um sistema ou um subsistema
  - Os testes se concentram nas interfaces de comunicação entre unidades
- Geralmente responsabilidade é de uma equipe independente de teste

Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Seção 2.2.3 Validação de Software
  - Cap. 8 Testes de Software

21 - Exercícios de Testes

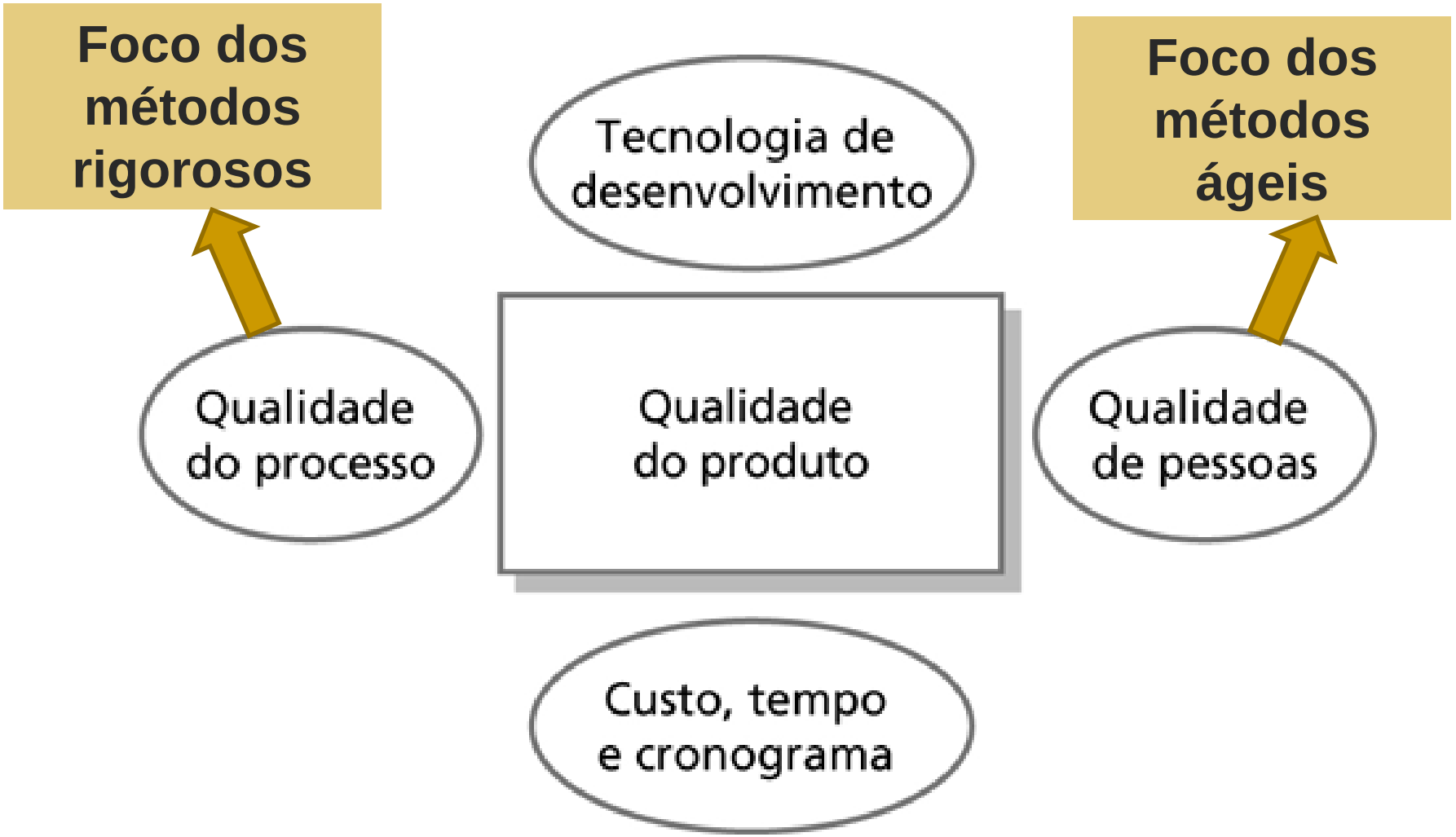
1. Criar testes automatizados (JUnit) para as funcionalidades implementadas na aula prática anterior

- Não precisa testar 100% do que foi implementado
  - Vou olhar principalmente os conceitos de OO e testes automatizados

22 - Qualidade de Software

- A qualidade de software tem se aprimorado nos últimos 15 anos
  - Empresas têm adotado novas técnicas
  - Orientação a objetos se difundiu
  - Ferramentas CASE têm sido usadas
- Na manufatura, qualidade significa atender às especificações
  - Em software, a definição não é tão simples

Qualidade de Software



Métricas de Software

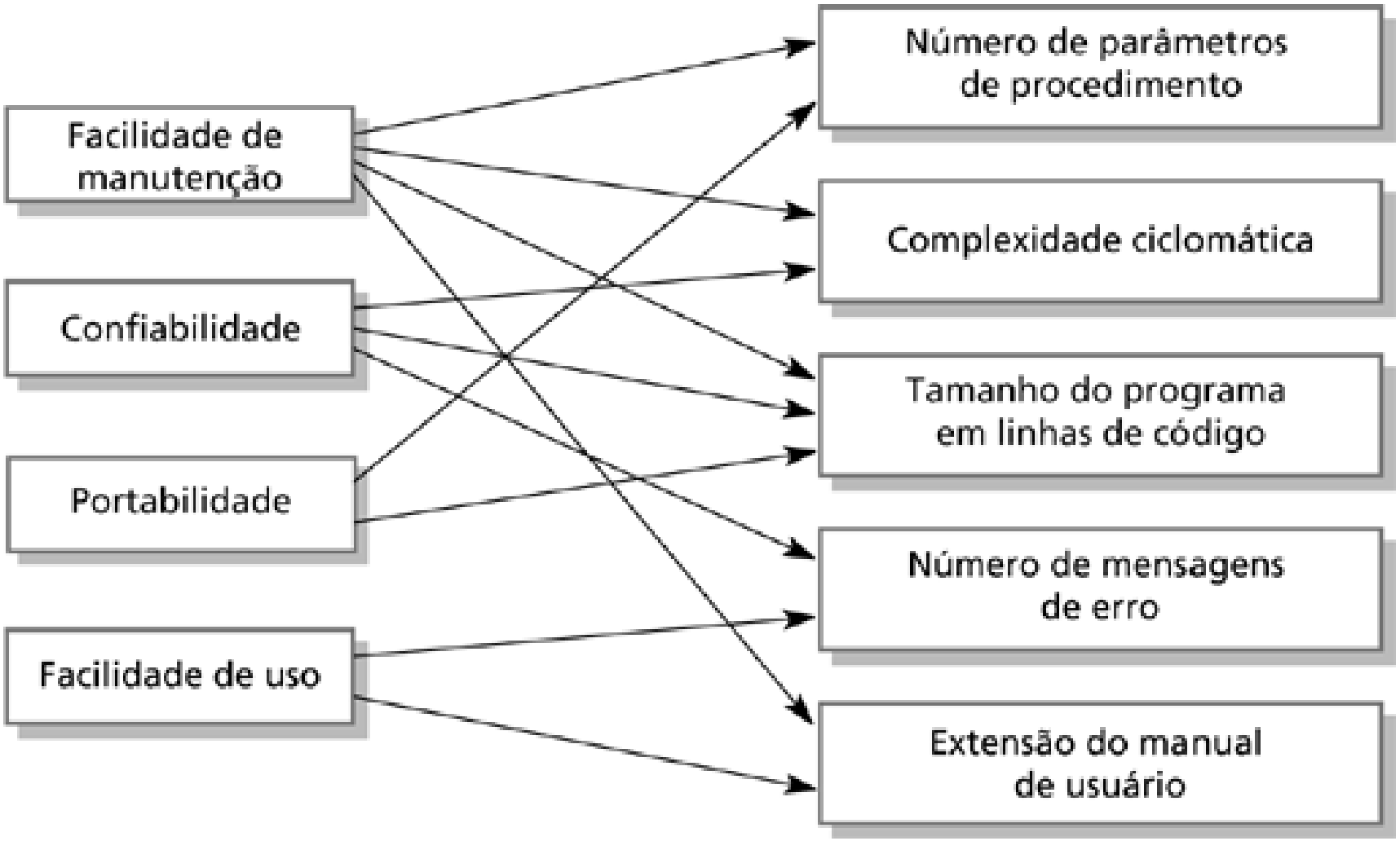
- Medições se dedicam a obter um ou mais valores numéricos para um atributo de qualidade

  - Ao comparar os números, é possível tirar conclusões sobre a qualidade do produto
- Uma métrica de software é qualquer medição que se refere ao sistema
  - Medições de tamanho (exemplo, LOC)
  - Número de defeitos relatados, etc.

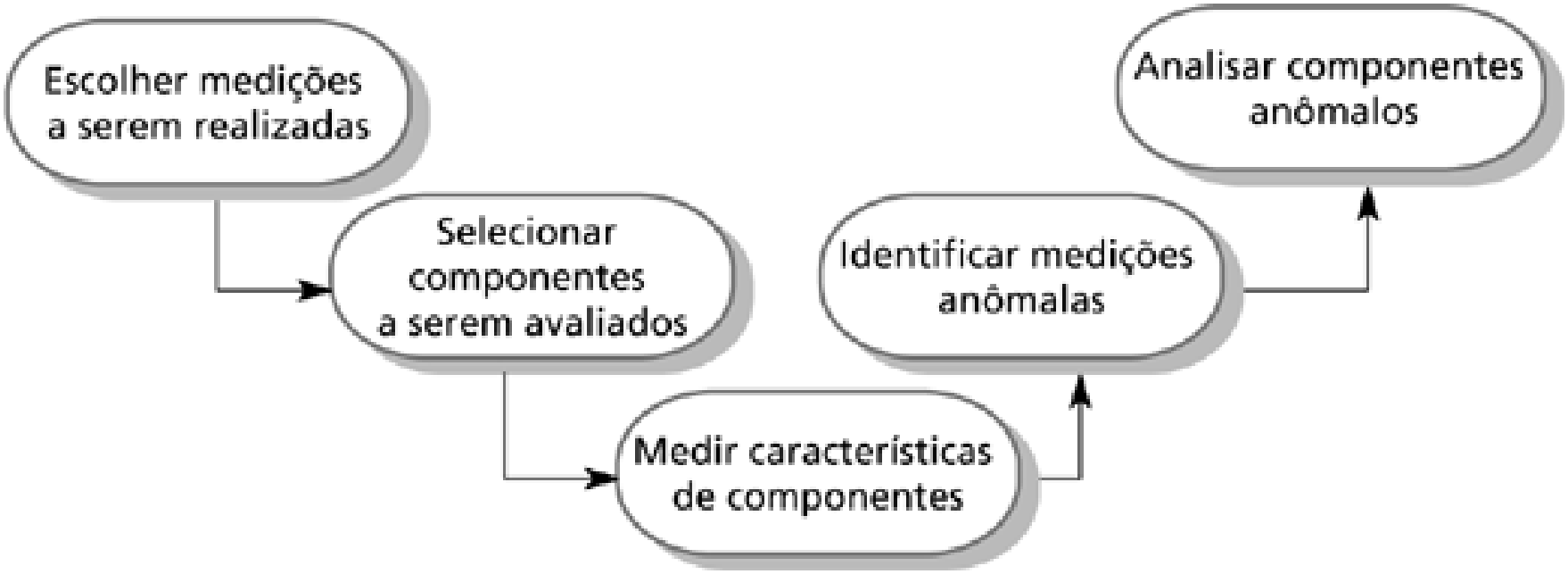
Modelos de Qualidade

- Relacionam atributos internos com atributos de qualidade
  - Atributos internos são mais facilmente quantificáveis
- Deveria haver um relacionamento claro e válido entre atributos de qualidade e atributos internos (ideal)

Um Modelo de Qualidade



Modelo do Processo



Métricas de Software

- Métricas Dinâmicas
  - São coletadas por medições realizadas durante a execução do programa (exemplo: tempo de execução)
- Métricas Estáticas
  - São coletadas por medições realizadas na documentação de projeto ou código fonte do programa (exemplo: linhas de código)

Métricas Estáticas

- Fan-in / Fan-out
- Tamanho do código
- Complexidade Ciclomática
- Tamanho do Vocabulário
- Profundidade de Aninhamento

Métricas de Programas OO

- Métricas de Chidamber-Kemerer (CK)
  - Métodos Ponderados por Classes (WMC)
  - Profundidade da Herança (DIT)
  - Número de Filhos (NOC)
  - Acoplamento entre Objetos (CBO)
  - Falta de Coesão em Métodos (LCOM)
- Número de Operações Sobreescritas

Bibliografia da Aula

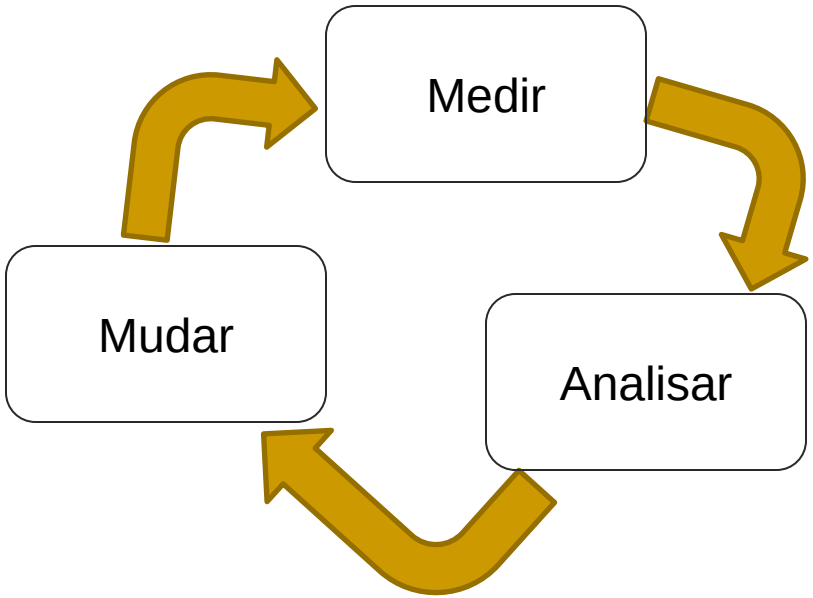
- Ian Sommerville. Engenharia de Software, 9ª Edição. Pearson Education, 2011.
  - Cap. 24 Gerenciamento de Qualidade

23 - Exercício de Medição

- Calcular métricas em artefatos de software
  - Diagramas arquiteturais
  - Diagramas de projeto detalhado
  - Código fonte

24 - Melhoria de Pocessos

- Processo é algo específico de uma organização
  - Não adianta tentar copiar o processo de outra empresa



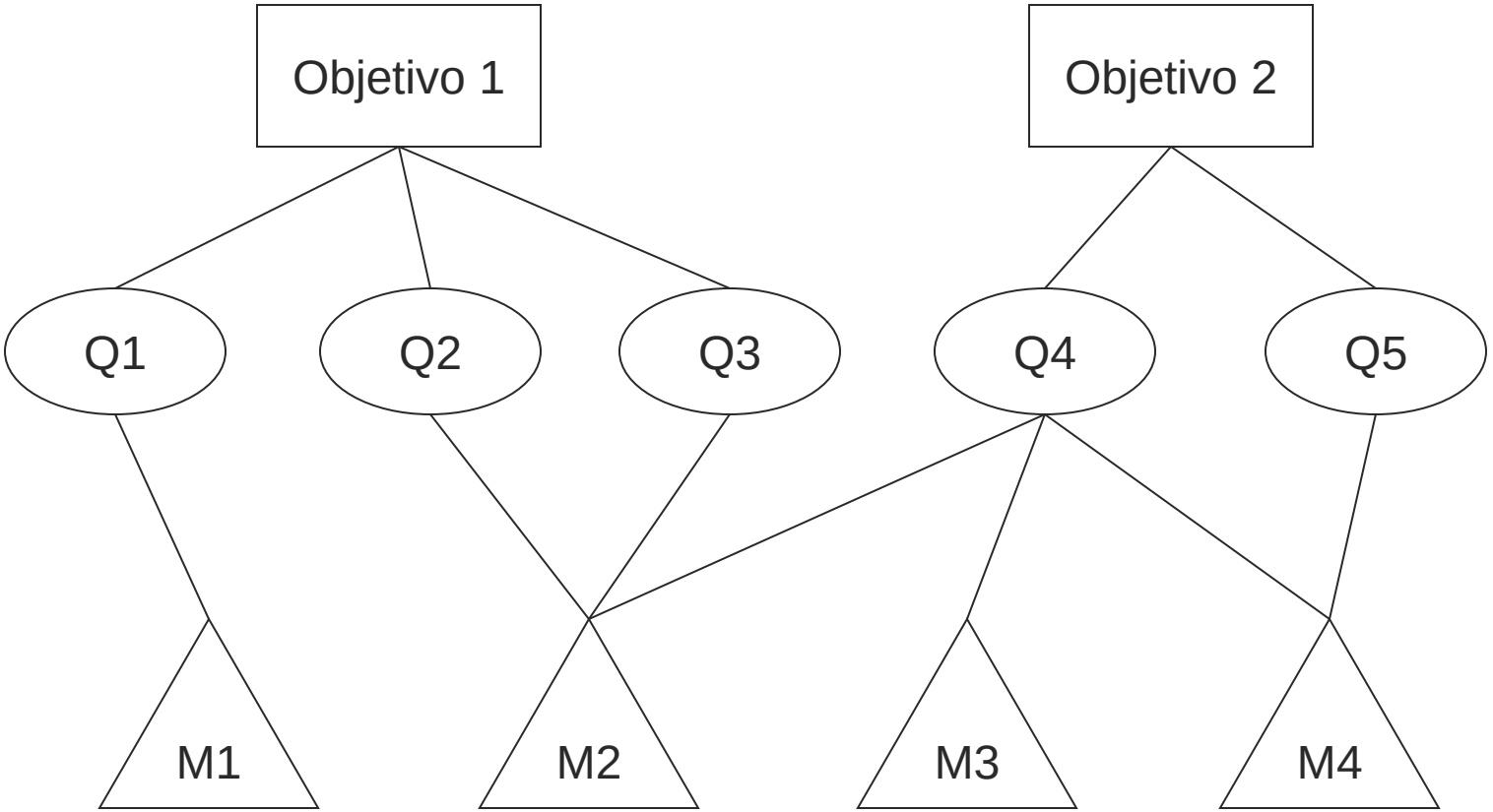
Medir Processo e Produto

- Medições podem ser usadas para quantificar ganhos de melhoria do processo
  - Exemplo: esforço e tempo dedicados a testes
- Medições de processo devem ser analisadas juntamente com medições de produto
  - Exemplo: robustez do produto (*bugs*)

Modelos de Medições

- Uma dificuldade fundamental é saber o que medir
  - Para isso, Basili propôs o modelo *Goal-Question-Metric (GQM)*
- Passos do GQM
  - Definir os objetivos
  - Formular as questões
  - Identificar as métricas

Representação do GQM



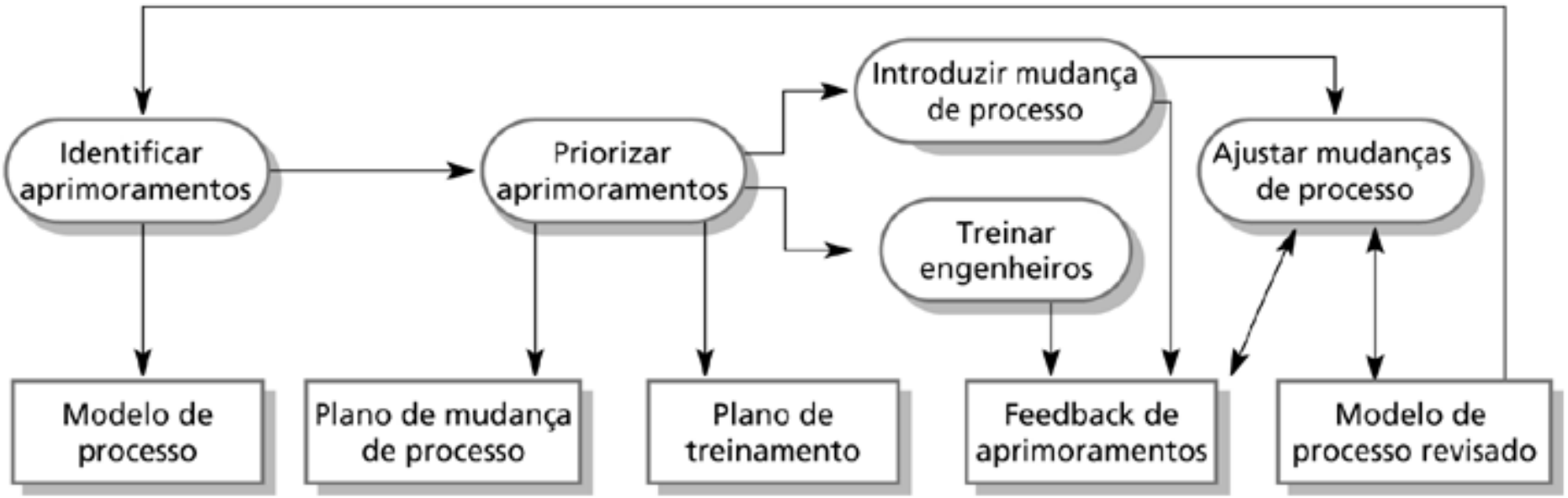
Análise do Processo

- A análise começa predominantemente qualitativa e passa a incorporar dados quantitativos
  - Fazem uso de medições
- A análise leva a mudanças do processo
  - Estudo de processos existentes
  - Desenvolvimento de um novo modelo

Mudança de Processo

- Realização de modificações no processo existente com base na análise
  - Introdução de novas práticas, métodos e ferramentas
  - Alteração de ordem das atividades do processo
  - Criação de alternativas para melhorar a comunicação
  - Criação de novos papéis ou responsabilidades, etc.

O Processo de Mudança



CMM Integrated (CMMI)

- Para tentar integrar os modelos de capacitação que foram surgindo, o SEI propôs o CMM Integrado (CMMI)

- O CMMI é complexo e define
  - Áreas de processos
  - Objetivos de cada área
  - Práticas para alcançar os objetivos

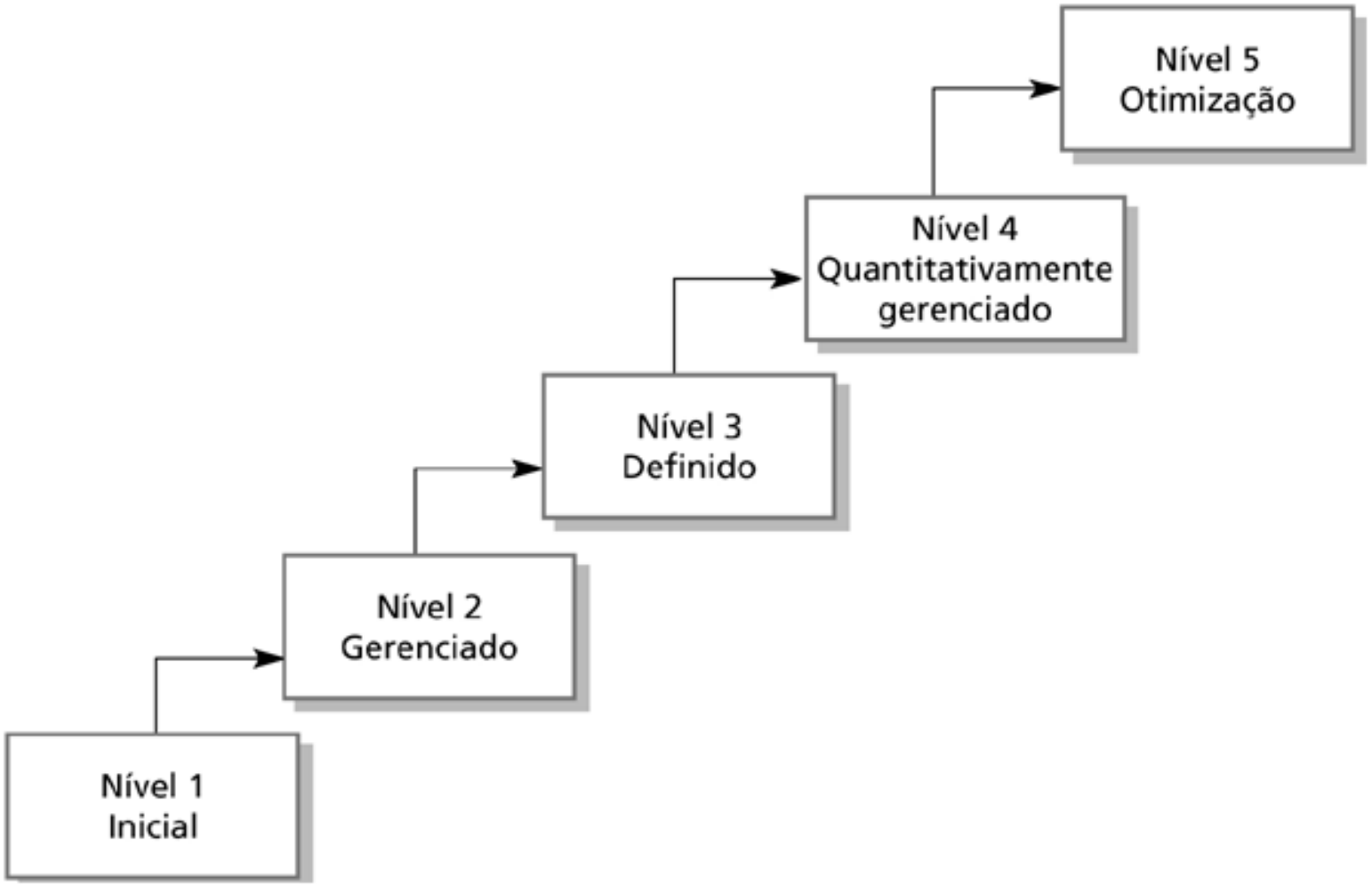
Áreas de Processos

- O CMMI define 22 áreas de processos
- As áreas de processo são organizados em quatro grupos
  - Gerenciamento de processos
  - Gerenciamento de projetos
  - Engenharia
  - Apoio

CMMI por Estágios

- Permite avaliar a capacitação do processo em cinco níveis
  - Descreve os objetivos que devem ser alcançados em cada nível de maturidade
- O aprimoramento do processo é atingido pela implementação das áreas de processo associadas a cada nível
  - As áreas permitem uma organização mover dos níveis mais baixos para os mais altos

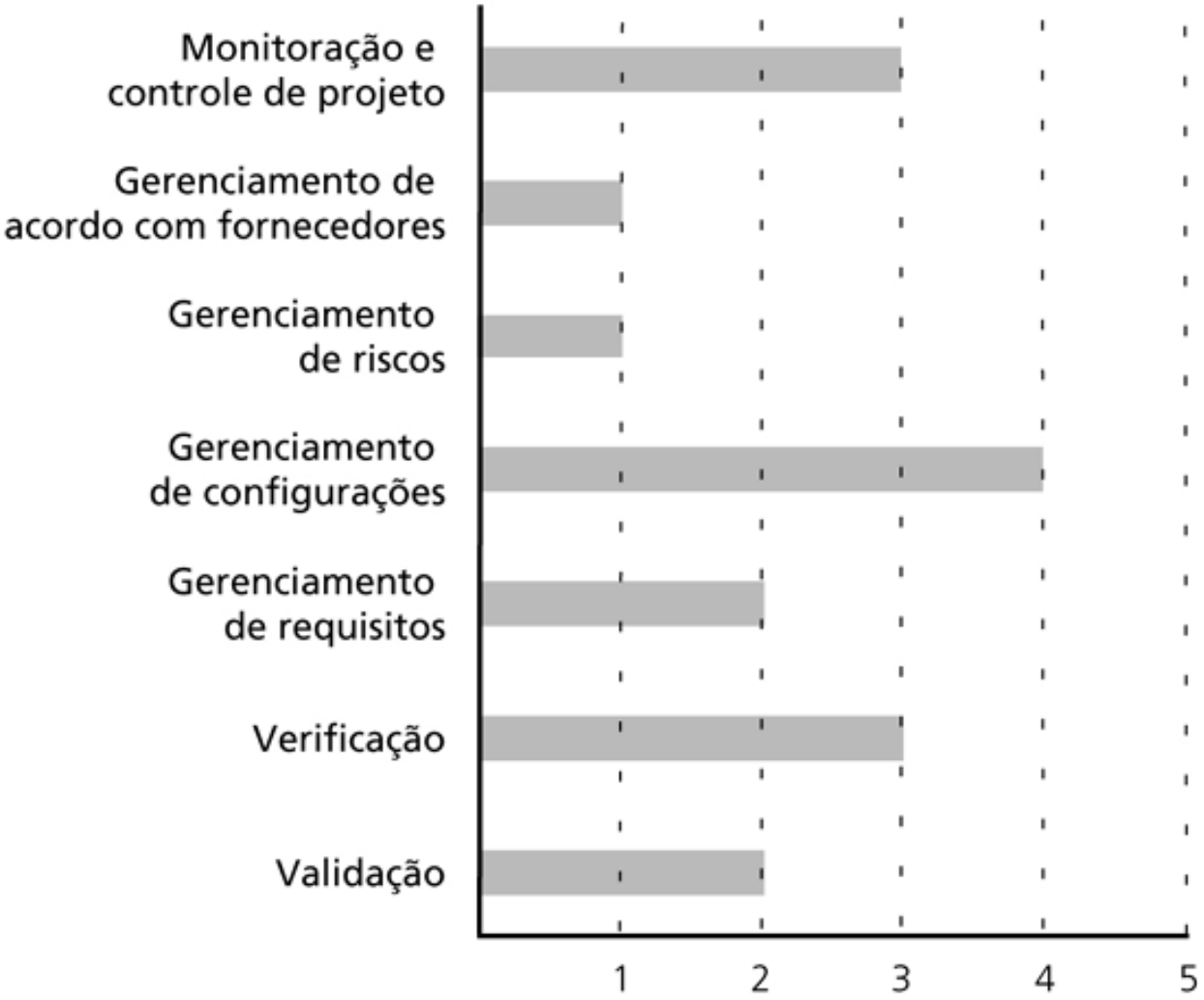
Os cinco níveis de maturidade



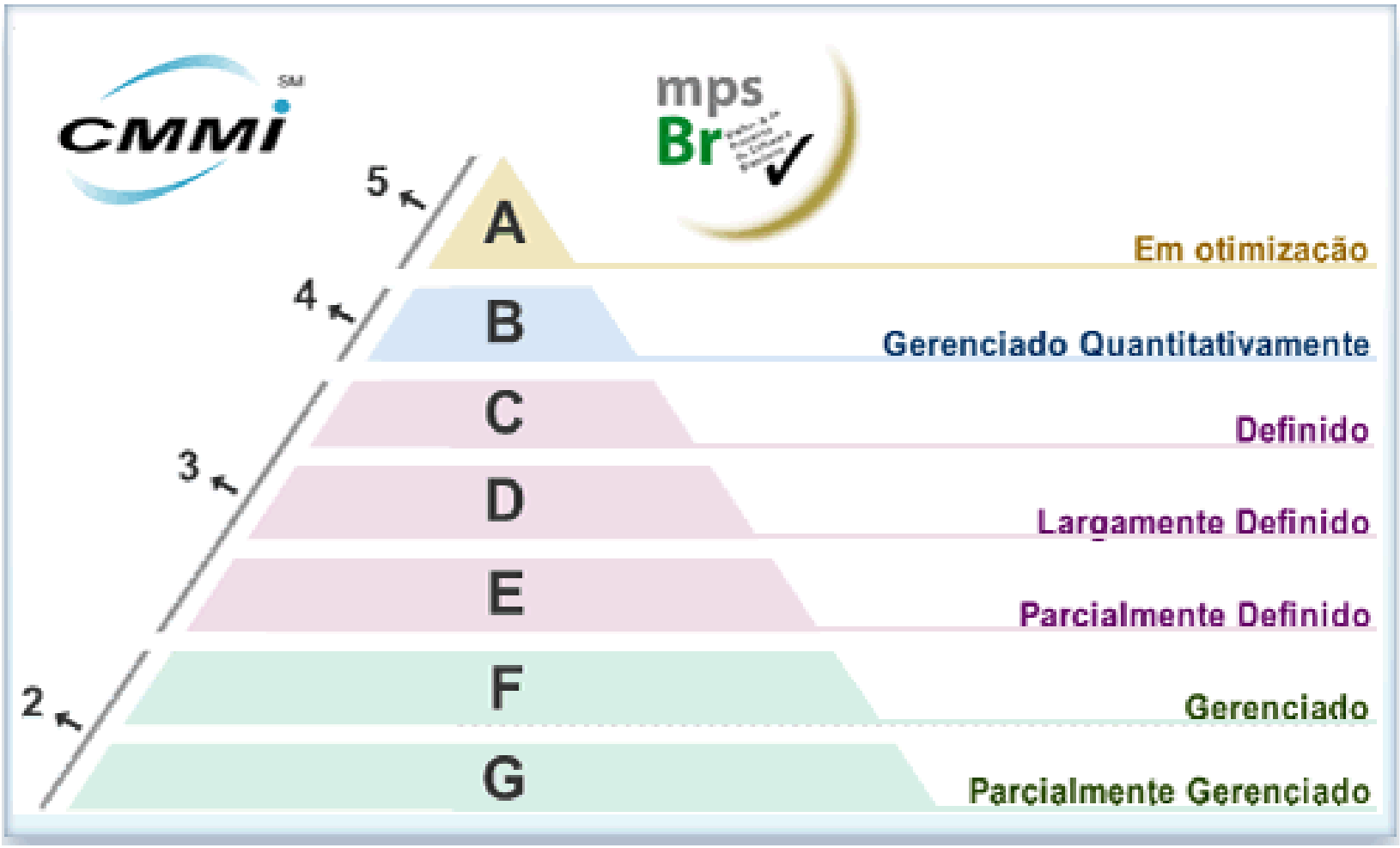
CMMI Contínuo

- Não classifica uma organização em níveis discretos
  - Considera as áreas de processo individualmente
- Motivação
  - As organizações operam em diferentes níveis de maturidade para as várias áreas de processo

Exemplo de Avaliação (parcial)



MPS.Br vs. CMMI



Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 26 Melhoria de Processos
- A. Koscianski e M. Soares. **Qualidade de Software**, 2ª Edição. Novatec, 2006.
  - Capítulo 7 MPS.Br