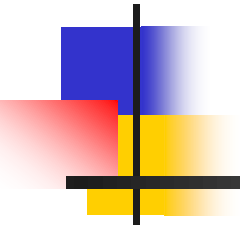
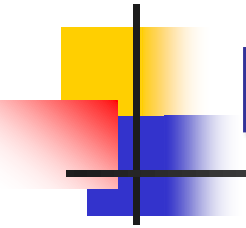


Decomposição, Modularização, Fatores de Qualidade de Software

Alberto Costa Neto

DComp - UFS

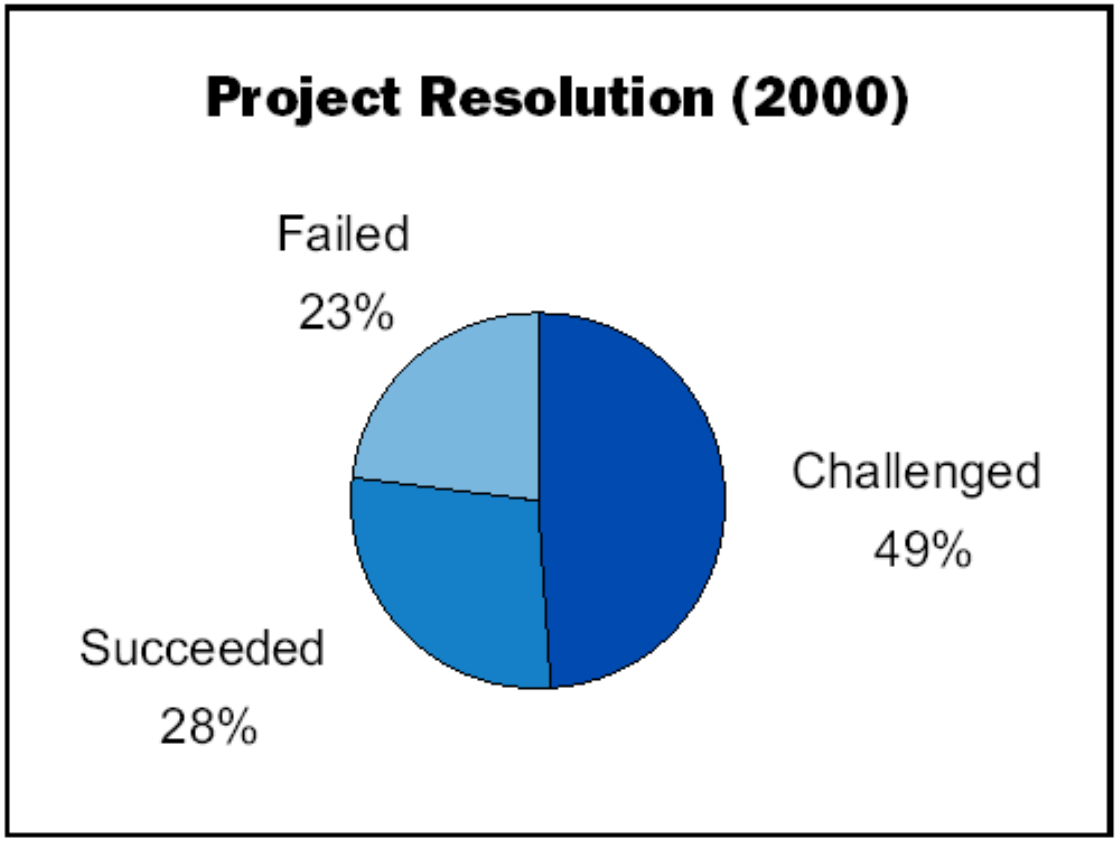


Roteiro

- Motivação
- Decomposição
- Projeto Modular Efetivo
- Fatores de Qualidade do Software

Motivação

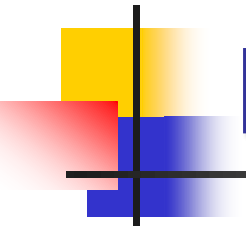
- A "Crise" do Software



Standish Group, Extreme Chaos

Motivação

- Quais as causas para a “crise” do software?
  - Falta ou não aplicação de métodos
  - Escassez de pessoal capacitado
  - Ausência de ferramentas de apoio
  - Complexidade do software



Roteiro

- Motivação
- **Decomposição**
- Projeto Modular Efetivo
- Fatores de Qualidade do Software

Decomposição

- Software é um produto complexo?
- Por quê?
  - Complexidade do domínio do problema
  - Flexibilidade do software
  - Instabilidade da tecnologia

Decomposição

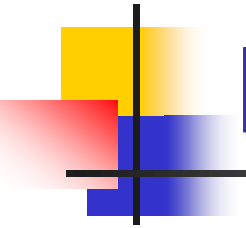
- Como lidar com a complexidade?
- Quanto ao Sistema Acadêmico?
  - É complexo?
  - Por quê?
  - Como vocês trabalhariam este sistema?

Decomposição

- A Engenharia de Software trabalha a complexidade por meio dos seguintes conceitos:
  - Abstração
  - Refinamento
  - Decomposição
    - Particionamento
    - Modularidade
  - Ocultamento da informação

Decomposição

- Abstração
  - É uma representação
  - Técnica poderosa para lidar com complexidade
  - Ignoramos detalhes que não são essenciais



Decomposição

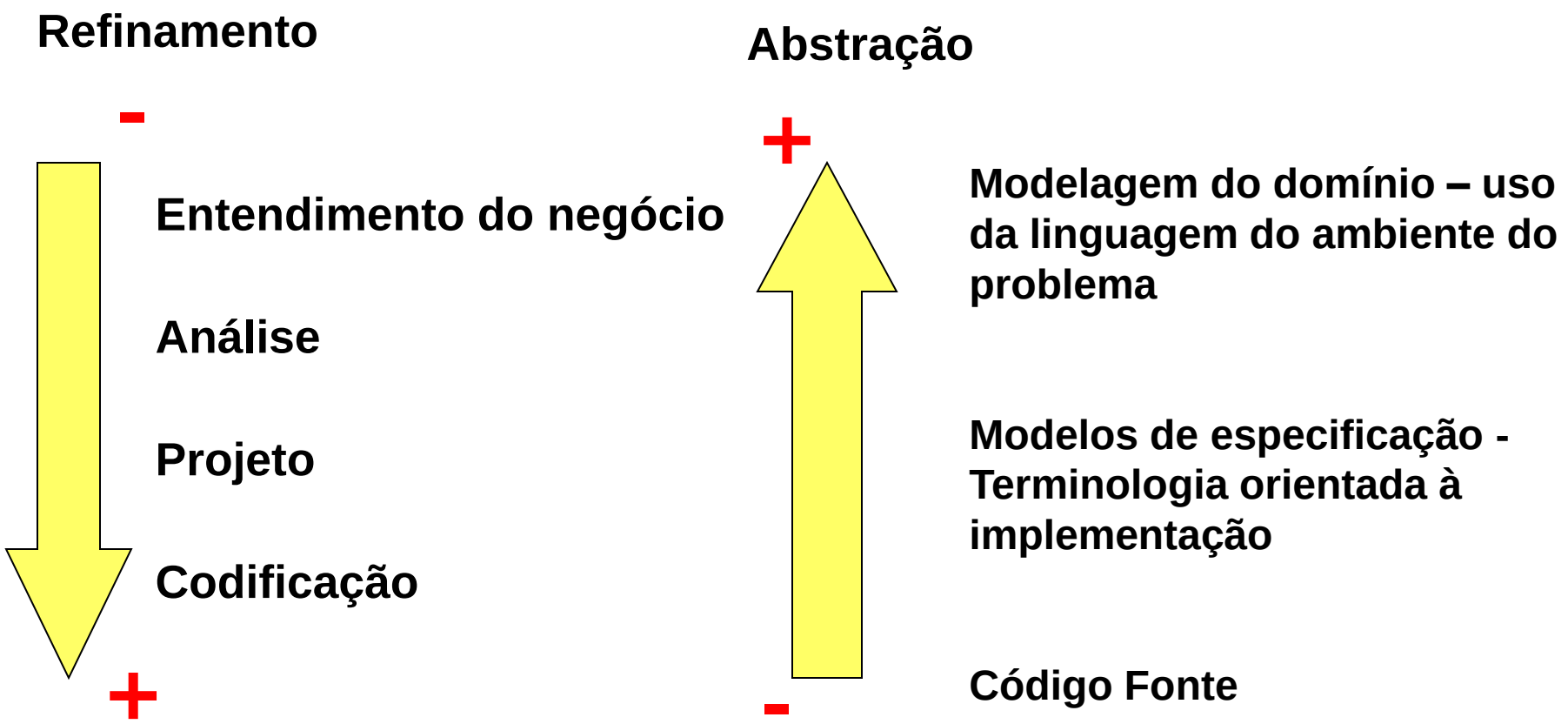
- Abstração
  - Exemplos de abstrações na programação
    - Linguagem de programação, abstrai linguagem de máquina
    - Procedimentos, abstraem comandos
    - Funções, abstraem expressões
  - Na OO, objetos são abstrações de entidades do mundo real

Decomposição

- Refinamento
  - Processo de detalhar uma abstração
  - Possibilita a definição de vários níveis de abstração

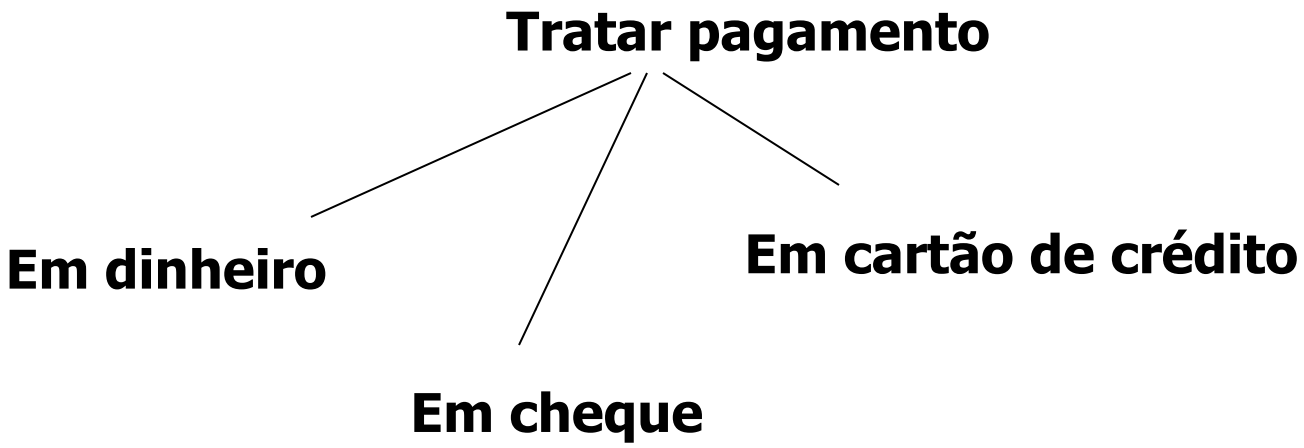
Decomposição

- Abstração e Refinamento são conceitos complementares



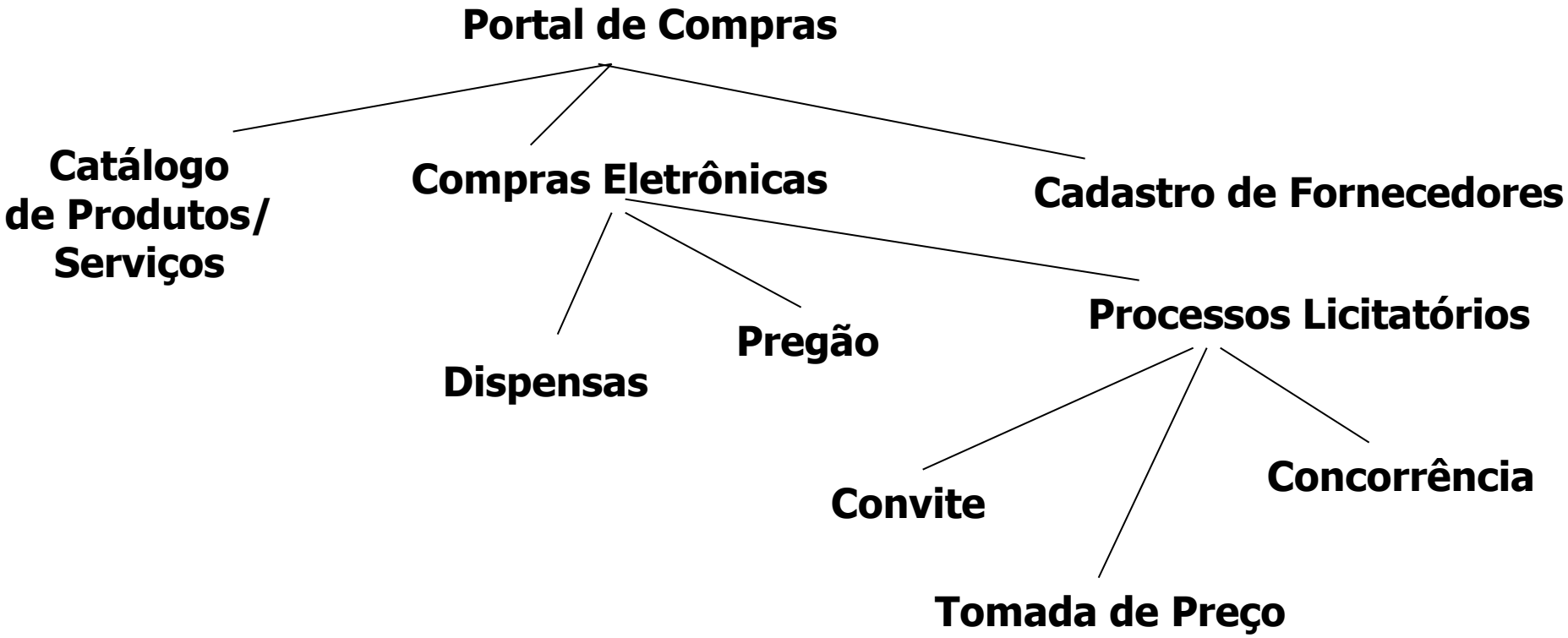
Decomposição

- Particionamento
  - Os problemas devem ser divididos em segmentos de modo que possam ser facilmente entendidos.



Decomposição

- Particionamento
  - Outro exemplo

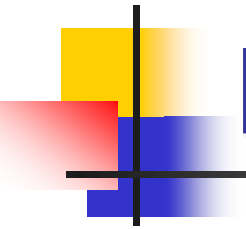


Decomposição

- Modularidade
  - Divide o software em componentes (módulos)
  - São nomeados e invocados
  - Objetivo: necessidade de se vencer a complexidade
    - Sistemas monolíticos são difíceis de compreender e manter
  - É mais fácil:
    - Complexidade (p1 + p2) > complex. (p1) + complex. (p2)
    - Esforço (p1 + p2) > esforço (p1) + esforço (p2).

Decomposição

- Modularização
  - Paradigma procedural
    - Módulos são procedimentos e funções
    - Fluxo de controle hierarquizado
  - Paradigma OO
    - Obtida a partir da definição das classes
    - Fluxo de controle colaborativo



Roteiro

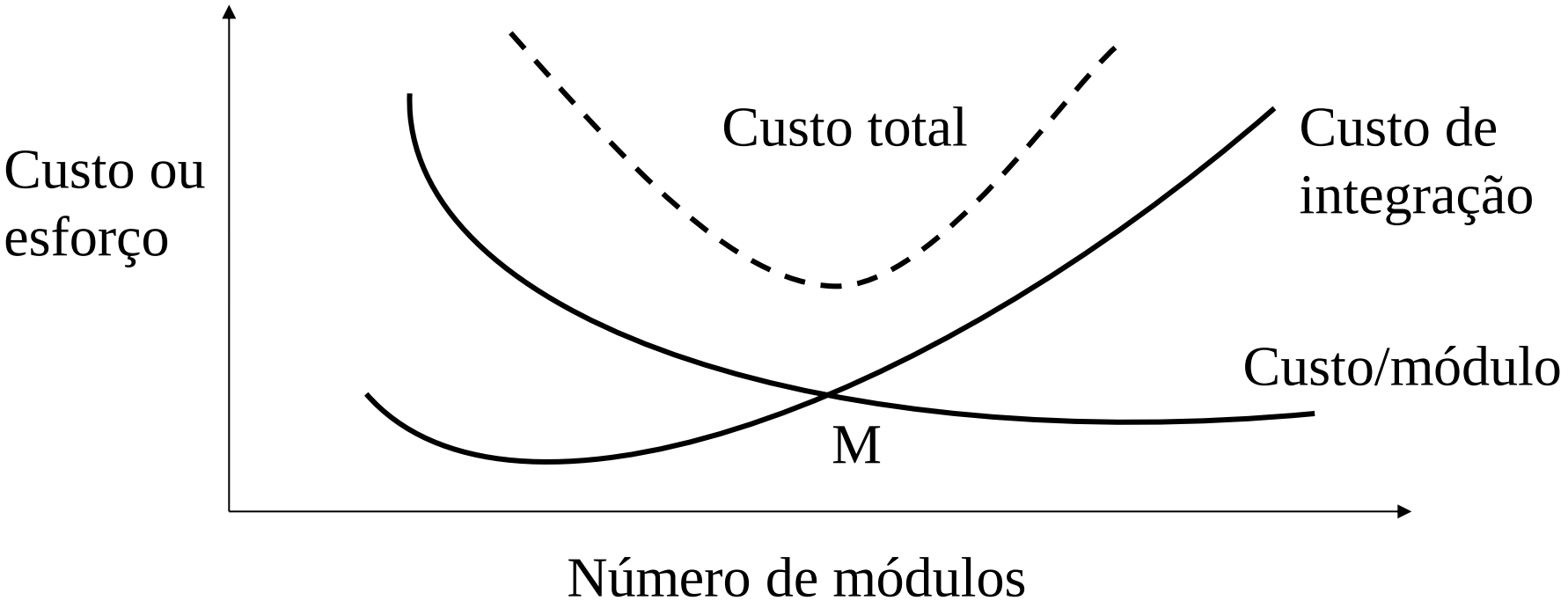
- Motivação
- Decomposição
- **Projeto Modular Efetivo**
- Fatores de Qualidade do Software

Projeto Modular Efetivo

- Características:
  - Modularização
  - Ocultamento da Informação
  - Independência Funcional
    - **Coesão**
    - **Acoplamento**

Projeto Modular Efetivo

- Modularidade



Projeto Modular Efetivo

- Vantagens da modularização:
  - Reduz a complexidade;
  - Possibilita o desenvolvimento em paralelo;
  - Facilita a modificação;
  - Possibilita a reutilização.

Projeto Modular Efetivo

- Ocultamento de Informação
  - Define e impõe restrições de acesso
  - Comunicação através da interface do módulo
  - Independência entre os módulos
  - Facilita a manutenção

Projeto Modular Efetivo

- Independência Funcional
  - Decorrência direta da modularidade e ocultamento da informação
  - Obtida quando o Módulo tem "finalidade única" e "aversão" a interação excessiva com outros módulos
  - É a chave para um bom projeto
  - Critérios qualitativos
    - Coesão
    - Acoplamento

Projeto Modular Efetivo

- Coesão
  - Um módulo coeso realiza uma única tarefa dentro de um procedimento de software
  - Indicação qualitativa do grau em que um módulo focaliza apenas uma coisa.

- Este código está coeso?

```
public void imprimeRecibo () {

        System.out.println ("**********************");
        System.out.println ("** Recibo do Cliente **");
        System.out.println ("**********************");

        double outstanding = 0.0;
        int index = 0;
        for (index = 0; index < _orders.size(); index++){
              outstanding += _orders.get(index).getAmount();
        }

        System.out.println (“Nome:" + _name);
        System.out.println (“Total:" + outstanding);
      }
}
```

- E agora?

```
public void imprimeRecibo () {

        imprimeCabecalho();

        double outstanding = calculaValorDevido();

        imprimeDetalhe(outstanding)
}

public void imprimeCabecalho() {
          System.out.println ("**********************");
          System.out.println ("** Recibo do Cliente **");
          System.out.println ("**********************");
}
```

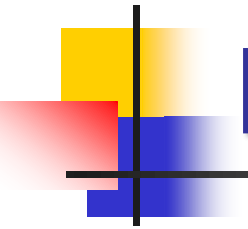
```
public double calculaValorDevido(){
        double outstanding = 0.0;
        int index = 0;
        for (index = 0; index < _orders.size(); index++){
            outstanding += _orders.get(index).getAmount();
        }
        return outstanding;
}

public void imprimeDetalhe(outstanding) {
        System.out.println ("Nome:" + _name);
        System.out.println ("Total:" + outstanding);
}
```

Projeto Modular Efetivo

- Acoplamento
  - Indicação qualitativa do grau com que um módulo está conectado a outros módulos e ao mundo exterior.
  - "Acoplamento diz respeito à independência entre componentes, a medida do impacto que a alteração da implementação de um componente tem sobre outros."

Projeto Modular Efetivo

- Princípios para a modularização
  - Unidades modulares linguísticas
    - Unidades sintáticas da linguagem usada
  - Poucas interfaces
  - Interfaces pequenas
  - Interfaces explícitas
  - Ocultamento da informação



Roteiro

- Motivação
- Decomposição
- Projeto Modular Efetivo
- **Fatores de Qualidade do Software**

Fatores de Qualidade

- Será que Cliente, Equipe de Desenvolvimento e Gerente têm a mesma opinião sobre a qualidade do software?

Fatores de Qualidade

- Fatores Externos
  - Rápido, confiável, fácil de usar, ...
  - Notados pelos usuários
- Fatores Internos
  - Modular, legível, fácil de modificar, ...
  - Perceptíveis pelos profissionais de computação
- Fatores internos são o caminho para alcançar os fatores externos

Fatores de Qualidade

- Funcionalidade
  - Grau em que o software satisfaz as necessidades declaradas
- Confiabilidade
  - "Probabilidade de um software operar livre de falhas , num ambiente especificado, durante um tempo especificado."
- Usabilidade
  - Esforço necessário para o uso do software por um usuário de perfil determinado

Fatores de Qualidade

- Eficiência
  - Grau em que o software faz uso otimizado dos recursos do sistema
- Manutenibilidade
  - Facilidade para fazer alterações
- Portabilidade
  - Facilidade do software ser transferido para outro ambiente