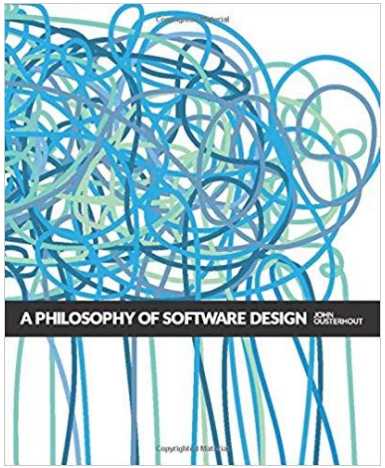
Introdução a Projeto de Software

Prof. Laerte Xavier

"O problema mais fundamental em Ciência da Computação é a tarefa de decomposição de problemas: como dividir um problema complexo em partes que possam ser resolvidas de forma independente" -- John Ousterhout



Projeto de Software

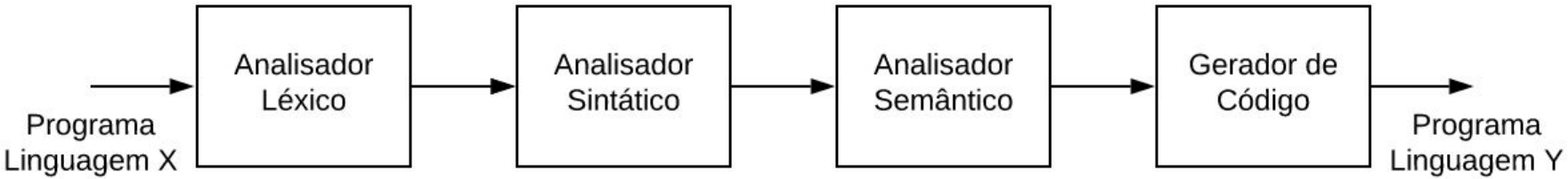
- Frase de Ousterhout é uma excelente definição
- Projeto:
  - Quebrar um "problema grande" em partes menores
  - Resolução (ou implementação) das partes menores resolvem (ou implementam) o "problema grande"

Project vs Design

- Em Inglês, temos duas palavras:
  - Project: esforço colaborativo para resolver problemas
  - Design: desenho ou proposta de uma solução
- Em Português, temos uma única palavra: projeto
- Neste módulo, **projeto = design**

Projetar = Quebrar em partes menores

- Exemplo: compilador

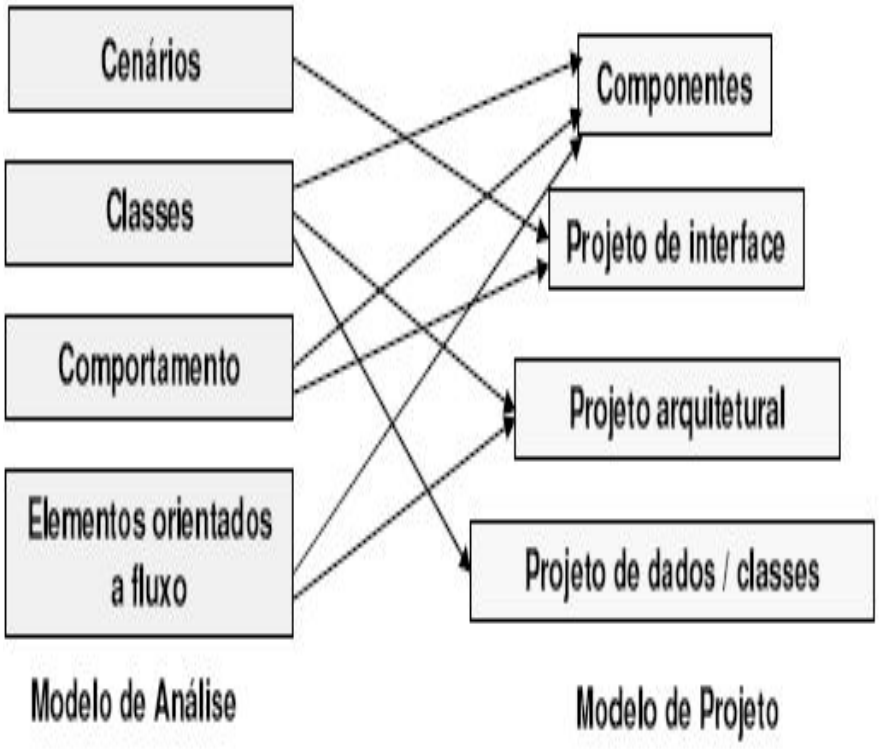


Projeto de Software

- Conjunto de princípios, conceitos e práticas que levam ao desenvolvimento de um software de alta qualidade:
  - Os princípios de projeto estabelecem uma filosofia que guia o trabalho que o projetista deve desempenhar
  - Os conceitos de projeto devem ser entendidos antes que a prática possa ser aplicada
  - A prática de projeto conduz à criação de várias representações do software que servem como guia para a construção que se segue

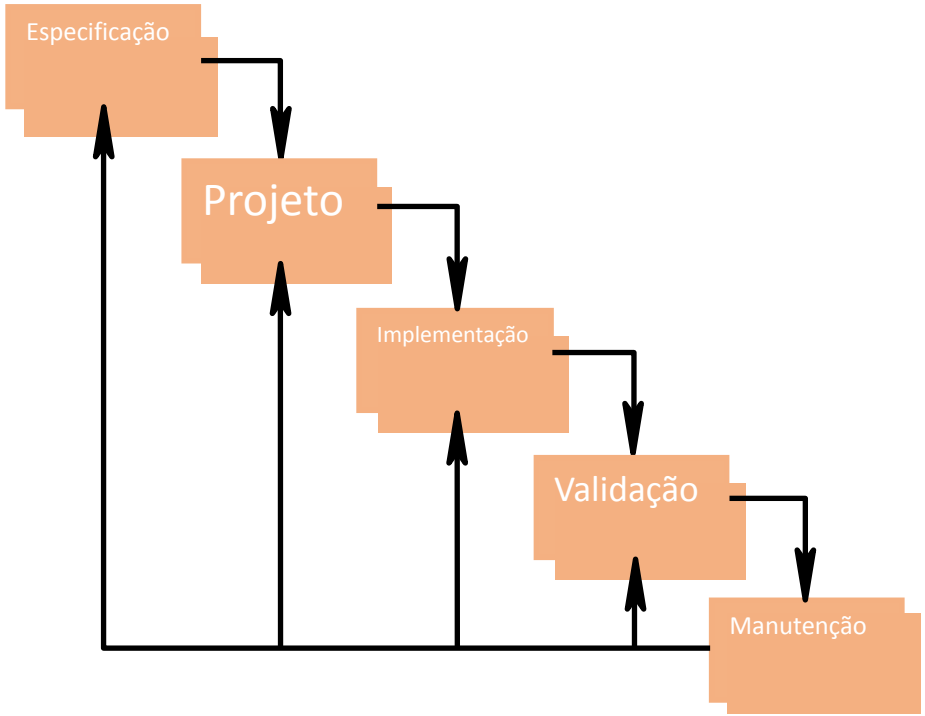
Projeto de Software

- Objetivo: produzir um modelo que satisfaça os requisitos.
  - Evolução contínua à medida que a análise se aperfeiçoa e entendimento amplia.



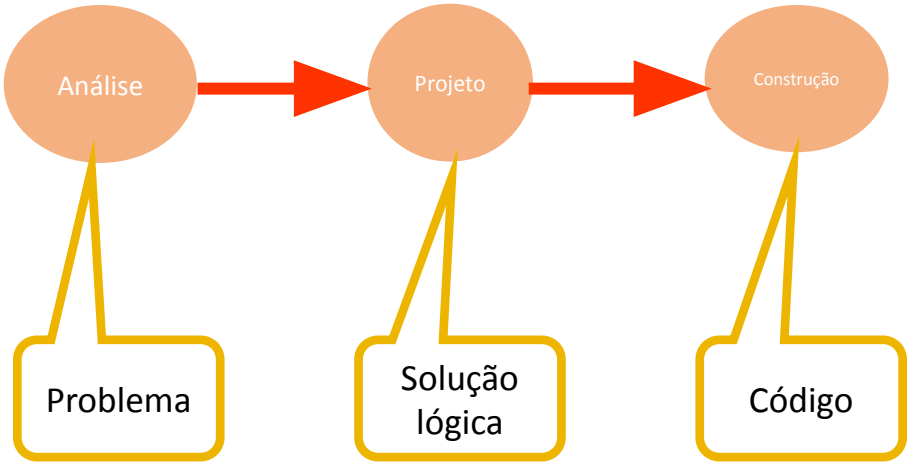
Projeto de Software na Engenharia de Software

- Projeto de software está no núcleo técnico da ES
- É aplicado a qualquer modelo de processo de software utilizado
- É a última ação da ES na atividade de modelagem
- Prepara o cenário para a construção

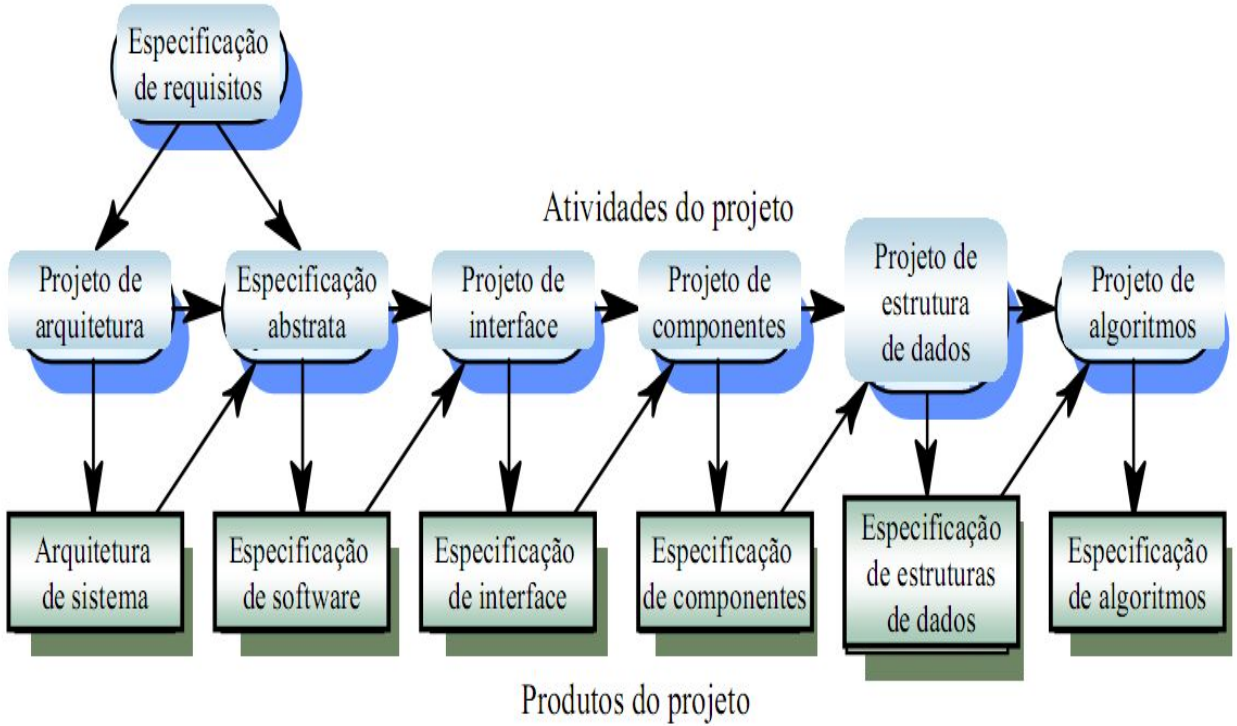


Da Análise ao Projeto

- Ambas são disciplinas de modelagem de criação e descrição de modelos
- Projeto de software é o modelo que é gerado quando são acrescentados aos requisitos funcionais os requisitos não-funcionais



Atividades de Projeto



Projeto de Software

- Qual é o produto do trabalho?
  - Um **modelo de projeto** que inclui representações de arquitetura, interface, componente e implantação.

- Como tenho certeza de que fiz corretamente?
  - O modelo de projeto é avaliado pela equipe de software em um esforço para determinar se contém erros, inconsistências ou omissões; se existem alternativas melhores e se o modelo pode ser implementado dentro das restrições, cronograma e custo que foram estabelecidos.

Projeto de Software

Projeto serve de base para todas as etapas da Engenharia de Software

- Projeto é necessário?
  - Sem projeto, como modificar, testar e avaliar a qualidade?
  - Permite avaliar a qualidade antes de implementar

Da Análise ao Projeto

As principais atividades realizadas na fase de projeto são:

1. Detalhamento dos aspectos dinâmicos do sistema.
2. Refinamento dos aspectos estáticos e estruturais do sistema.
3. Detalhamento da arquitetura do sistema.
4. Definição das estratégias para armazenamento, gerenciamento e persistência dos dados manipulados pelo sistema.
5. Realização do projeto da interface gráfica com o usuário.
6. Definição dos algoritmos a serem utilizados na implementação.

Motivações para Elaborar o Projeto

- Limitações da Tecnologia – Tecnologia Imperfeita
  - Custo
  - Capacidade de armazenamento
  - Velocidade de processamento
  - Aptidão dos processadores
  - Banda links de comunicação
  - Falibilidade

Motivações para Elaborar o Projeto

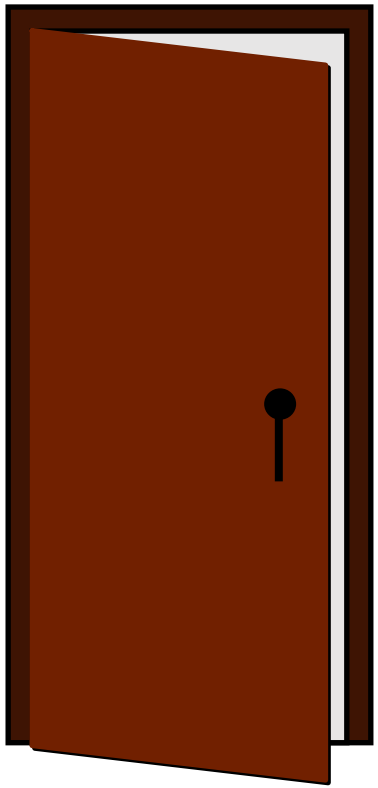
- Impactos da tecnologia imperfeita nos projetos
  - Fragmentação
  - Redundância
  - Convolução (servem atividades essenciais)
  - Conglomeração
  - Imensidão (localização geográfica)

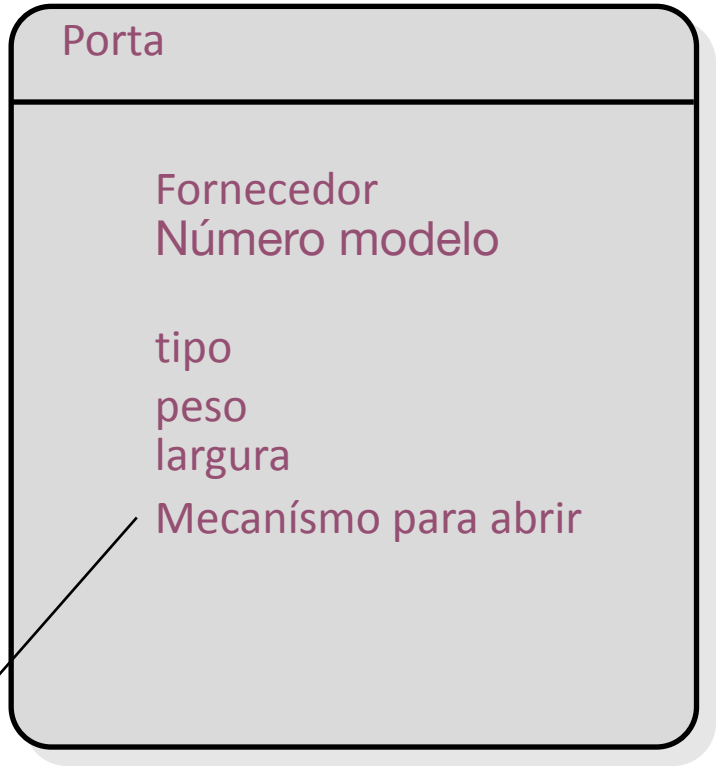
Conceitos Básicos Aplicados a Projetos

Abstração

- Ao se considerar uma solução modular para qualquer problema, muitos níveis de abstração podem se apresentar
- Abstração permite que nos concentremos no problema em algum nível de generalização sem considerar detalhes
- cada passo no processo de desenvolvimento de um sistema é um refinamento no nível de abstração da solução

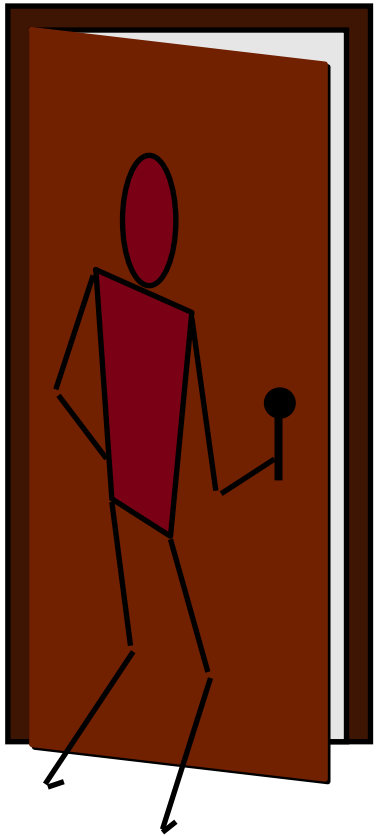
Abstração de Dados

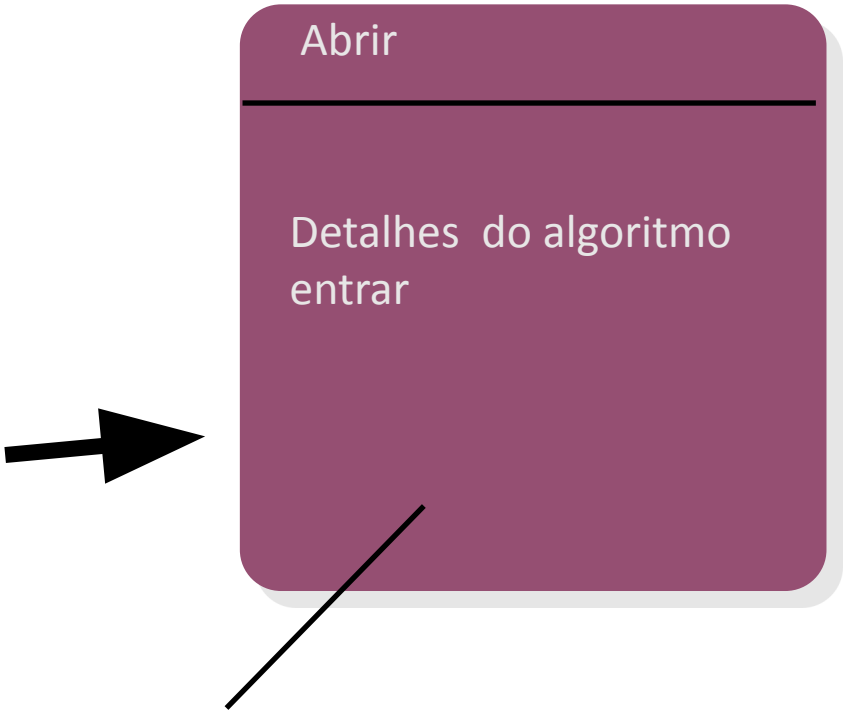




Implementado como estrutura de dados

Abstração Procedural





implementado com um "conhecimento"
do objeto que está associado com "entrar"

Arquitetura

- Arquitetura de software refere-se à “organização geral do software aos modos pelos quais disponibiliza integridade conceitual para um sistema”
  - É a estrutura ou a organização de componentes de programas, a maneira através da qual esses componentes interagem e a estrutura de dados que são usadas pelos componentes

- Conjunto de propriedades que devem ser especificadas como parte de um projeto de arquitetura
  - Propriedades estruturais
  - Propriedades não funcionais
  - Famílias de sistemas

Padrões

- Padrão e parte de um conhecimento consolidado já existente que transmite a essência de uma solução comprovada para um problema recorrente em certo contexto

- A descrição de um padrão permite ao projetista determinar se o padrão se aplica ou não ao trabalho em questão e se ele pode ser ou não utilizado

Separação por Interesses

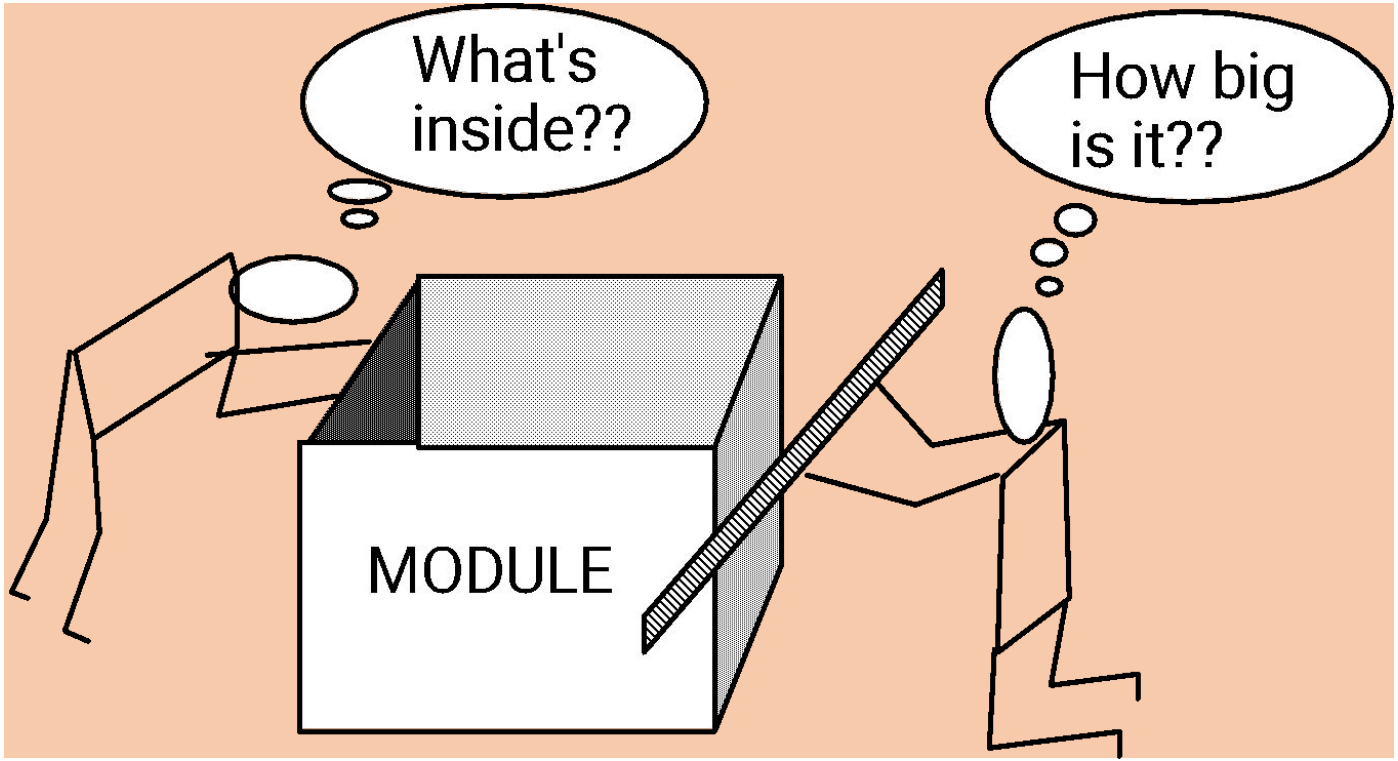
- Sugere que qualquer problema complexo pode ser tratado mais facilmente se for subdividido em partes para serem desenvolvidos independentemente

- A complexidade percebida de dois problemas, quando estes são combinados, normalmente é maior do que a soma da complexidade percebida quando cada um deles é tomado individualmente

Modularidade

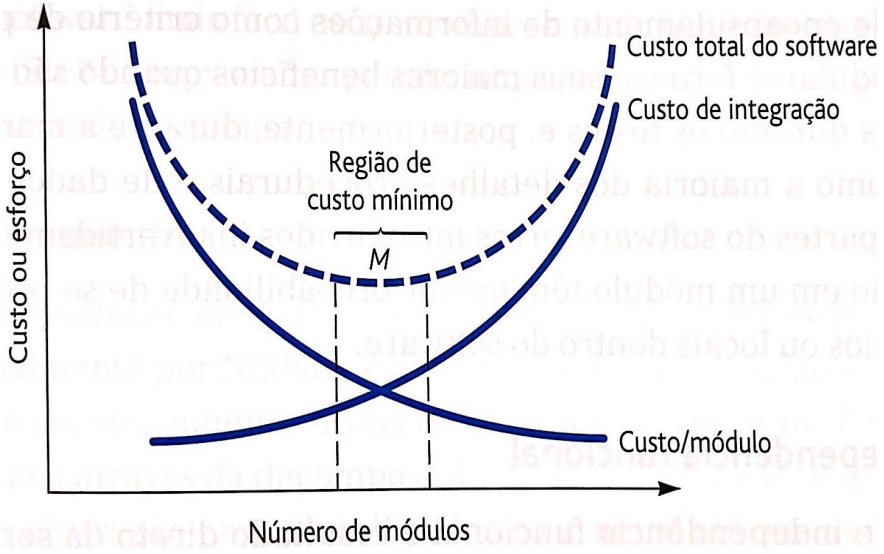
- Característica dos sistemas que são divididos em componentes
- Modularização permite que um sistema complexo seja compreendido pela mente humana (não consegue lidar com toda a complexidade dos sistemas de uma só vez)
- Abstração permite que nos concentremos no problema em algum nível de generalização sem considerar detalhes
- Pergunta: Quantos módulos e qual o tamanho ideal dos módulos?

Tamanho do Módulo: duas visões



Modularidade

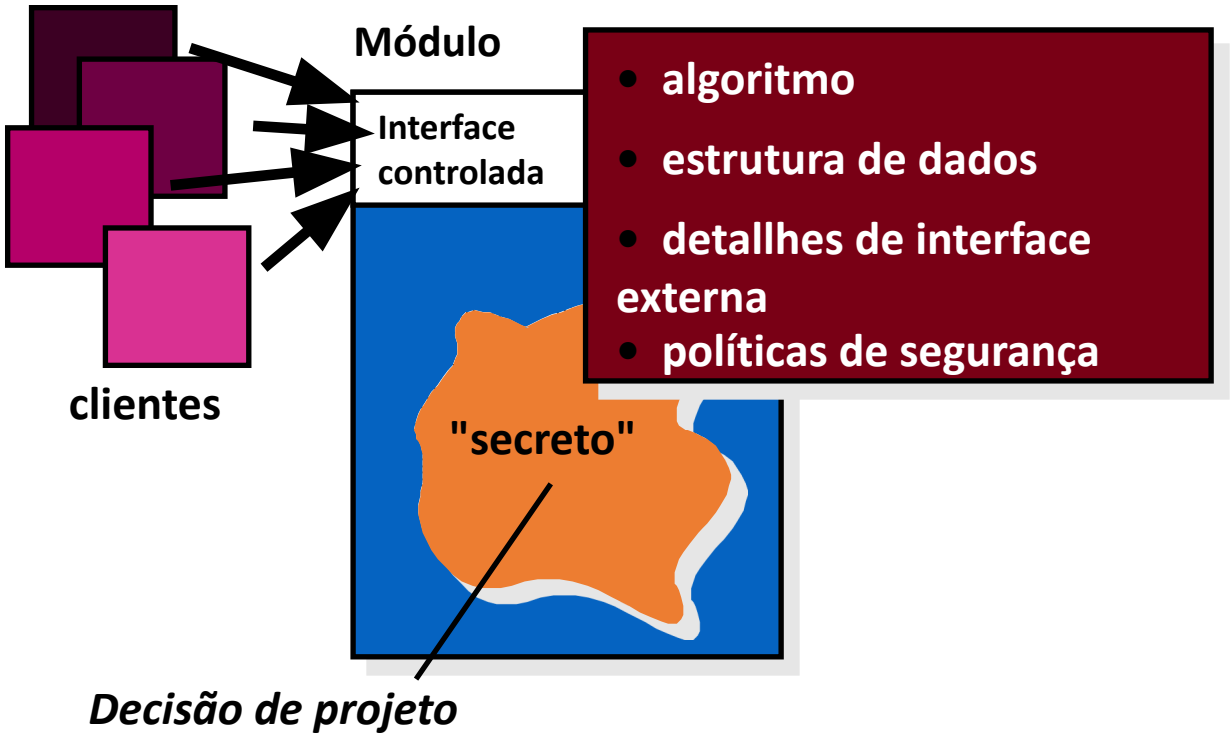
- Módulos são integrados com o objetivo de atender a um requisito
- “Dividir e conquistar”
- ***What is the "right" number of modules for a specific software design?***



Ocultação de Informação

- O princípio da *ocultação de informações* sugere que cada módulo deve conter decisões de projeto que ele oculta dos outros módulos
- Módulos comunicam entre si para obterem informações necessárias para o que o sistema funcione

Ocultação de Informação



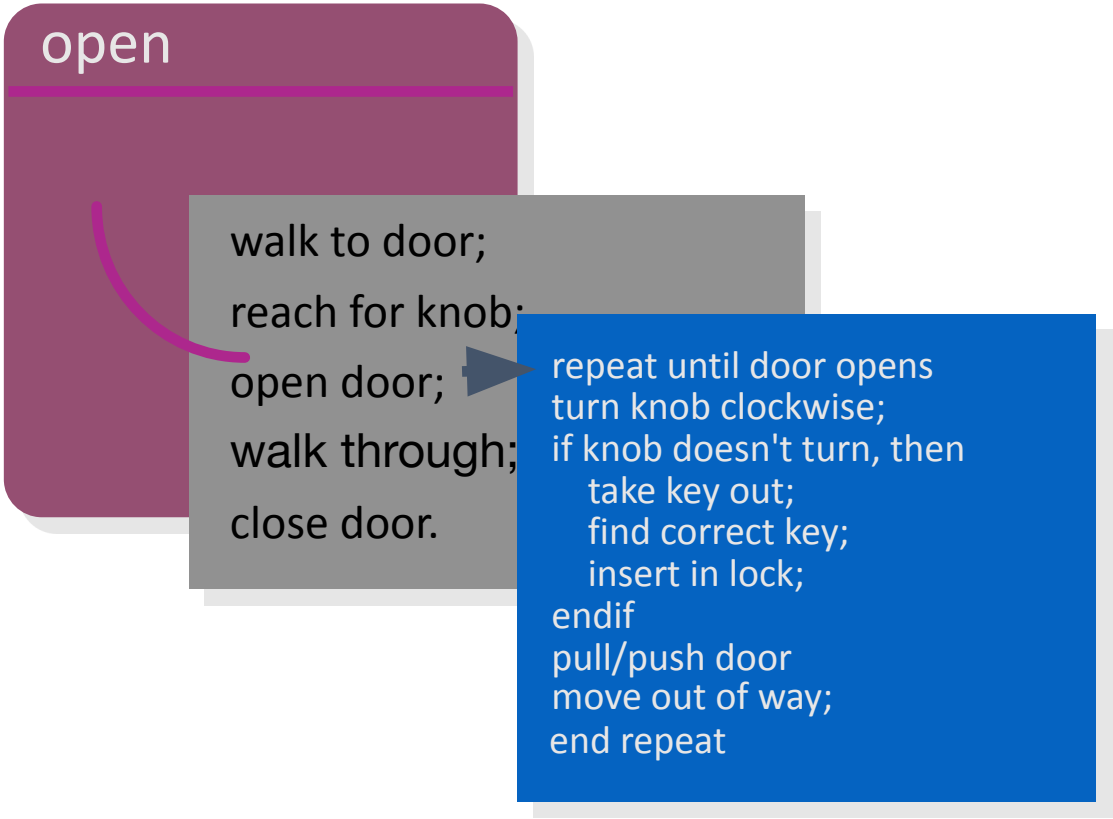
Independência Funcional

- A independência funcional é alcançada através do desenvolvimento de módulos simples e uma "aversão" à interação excessiva com outros módulos.
- Coesão é uma indicação da força relativa funcional de um módulo.
- Acoplamento é uma indicação da relação entre os módulos interdependência.

Refinamento

- Uma elaboração maior no nível de detalhes
- Um programa é desenvolvido através de sucessivos refinamentos
  - chegamos a uma descrição em linguagem de programação partindo de uma em linguagem natural e refinando-a em sucessivas iterações
- Abstração e refinamento são conceitos complementares

Refinamento



Refatoração

- Fowler define refatoração da seguinte maneira:
  - "Refatoração é o processo de mudança de um sistema de software de tal maneira que não altere o comportamento externo do código [design] e ainda melhora a sua estrutura interna."
- Quando o software é reformulado, o projeto existente é examinado para verificar:
  - redundância
  - elementos de design não utilizados
  - algoritmos ineficientes ou desnecessários
  - estruturas de dados inadequadas ou mal construída
  - ou qualquer outra falha de design que pode ser corrigido para produzir um design melhor..

Independência Funcional

- Modularidade + abstração + ocultação = Independência Funcional

  - “Finalidade única” e menos interação
  - Interfaces simplificadas
  - Manutenção mais fácil
  - Propagação de erros minimizada
  - Reutilização

  - Dois critérios (qualitativos) para avaliação
    - COESÃO: robustez funcional de um módulo (módulo realiza uma única tarefa)
    - ACOPLAMENTO: indicação da interdependência entre módulos

Refinamento e Refatoração

- Refinamento
  - Processo de elaboração (alto nível -> mais detalhes)
  - Refinamentos sucessivos
  - (Abstração + refinamentos): conceitos complementares

- Refatoração
  - Reorganizar para simplificar o projeto sem alterar as funções e os comportamentos.
  - O que pode ser refatorado?
    - Redundância, elementos não utilizados, algoritmos ineficientes, etc

Princípios e Boas Práticas de Projeto

Princípios Básicos de Projetos

- **Divisão de responsabilidades:** Divida seu aplicativo em recursos distintos com o pouca sobreposição na funcionalidade
- **Princípio da responsabilidade única:** Cada componente ou módulo deve ser responsável para apenas um recurso ou uma funcionalidade específica ou agregação de funcionalidade coesa
- **Princípio do Conhecimento Mínimo (também conhecido como Lei de Demeter):** Um componente ou objeto não deve saber sobre detalhes internos de outros componentes ou objetos.

Princípios Básicos de Projetos

- **Não se repita (Don’t repeat yourself):** Uma funcionalidade em um único local
- **Minimize o design inicial:** Apenas projete o que for necessário

Boas Práticas de Projetos

- Mantenha padrões de projeto consistentes dentro de uma mesma camada
  - Mantenha o design de componentes consistente para uma determinada operação dentro de uma camada lógica
  - Avaliar quando a complexidade e variação dos requisitos é muito alta
- Não duplique funcionalidades
  - Aumenta a coesão
  - Facilita a otimização e manutenção
- Estabeleça um estilo de codificação e uma convenção de nomenclatura
  - Facilita o entendimento
  - Otimiza a manutenção

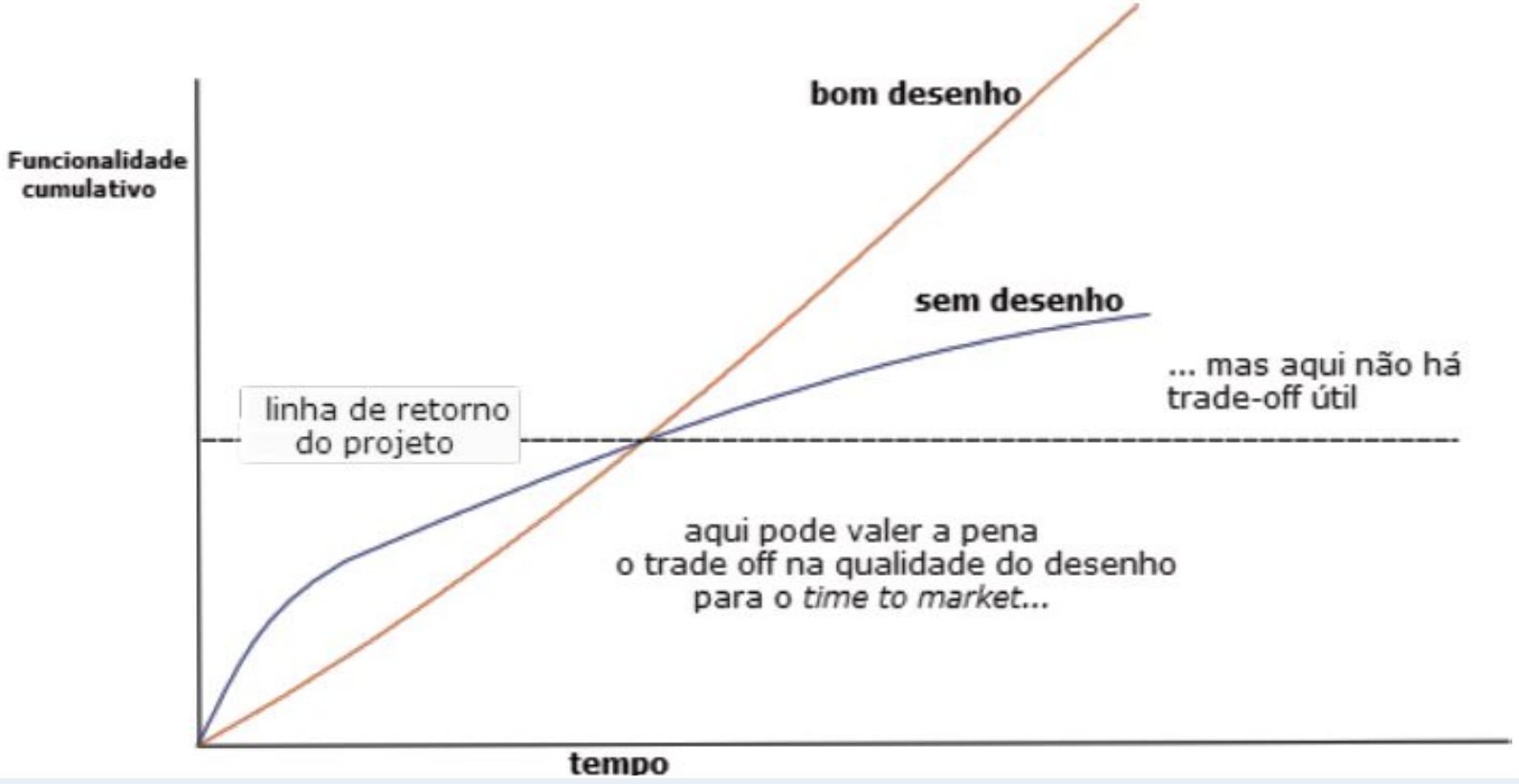
Boas Práticas de Projetos

- Considere abordagens alternativas
  - Padrões
  - Estilos de arquitetura
  - Tecnologias
- Ser rastreável ao modelo de análise
  - Consistência entre modelos os artefatos
  - Facilita o entendimento
- Minimize a distância entre o software e o problema do mundo real
  - Software inovador
  - Diferença para o negócio

Boas Práticas de Projetos

- Organize-se para gestão da qualidade
  - Estruture para acomodar mudanças
  - Use técnicas e ferramentas automatizadas de controle de qualidade
  - Use testes unitários, análise de dependência, análise de código estático ...
  - Defina métricas comportamentais e de desempenho para componentes e subsistemas
- Considere os aspectos operacionais da aplicação
  - Determine métricas operacionais
  - Estabeleça níveis de dependências

Bom Projeto é Investimento!



FOWLER,M. Design Stamina Hypothesis

Conclusão

- Projeto é o que quase todo engenheiro de software quer fazer
- É o lugar onde a criatividade impera

Exercícios de Fixação

Com relação aos princípios aplicados ao projeto de software, relacione a segunda coluna de acordo com a primeira e identifique abaixo a sequência correta para os relacionamentos feitos.

Princípio		**Descrição**
A. Modularização		Sugere que cada módulo deve conter decisões de projeto que ele não precisa mostrar aos outros módulos.
B. ocultação de informações		Permite que nos concentremos no problema em algum nível de generalização sem considerar detalhes
C. Abstração		Parte de um conhecimento consolidado já existente que transmite a essência de uma solução comprovada para um problema recorrente em certo contexto.
D. Refinamento		Permite que um sistema complexo seja compreendido a mente humana não consegue lidar com toda a complexidade dos sistemas de uma só vez.
E. Padrão		Permite chegar a uma descrição em linguagem de programação, por exemplo, partindo de uma em linguagem natural e refinando-a em sucessivas iterações.

Qual princípio de projeto é violado pelo seguinte código?

```
 void imprimeDataContratacao(Funcionario func) {
   Date data = func.getDataContratacao();
   String msg = data.format();
   System.out.println(msg);
}
```

Qual princípio de projeto é violado pelo seguinte código?

```
 void sendMail(ContaBancaria conta, String msg) {
   Cliente cliente = conta.getCliente();
   String endereco = cliente.getMailAddress();
   "Envia mail"
}
```