DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Desenvolvimento de *software*

Ivan Luiz Marques Ricarte

2002

`http://www.dca.fee.unicamp.br/~ricarte/`

Sumário

- O papel do *software* no trabalho acadêmico
- Evolução do desenvolvimento de *software*
- Por que desenvolver *software* é difícil?
- Como desenvolver bom *software*?
- O papel da orientação a objetos

O papel do *software* no trabalho acadêmico

- Do ponto de vista do trabalho:
  - Validação de hipóteses
  - Novos caminhos para soluções
- Para o aluno:
  - Aprender a desenvolver *software* de qualidade

Evolução do desenvolvimento de *software*

1950–60's *Software* orientado pelo *hardware*

1960–70's *Software* como produto
bibliotecas de *software*

1970–80's *Software* em sistemas complexos
popularização de microprocessadores
sistemas distribuídos

1990's–?? *Software* responsável pela maior parte do custo em sistemas computacionais

Por que desenvolver *software* é difícil?

- Frederick Brooks: *No Silver Bullet* para construir *software*
  - Conjunto de construções conceituais
  - Complexo e não-linear
  - Sujeito a mudanças e modificações
  - Invisível e não-visualizável
- Phillip Armour: *software* não é um produto, mas uma forma de armazenar conhecimento
  - desenvolver *software* não é "produzir um produto", mas adquirir conhecimento

Disciplina no desenvolvimento de *software*

- O foco na qualidade em desenvolvimento de *software* depende da aplicação consistente e disciplinada de

  processos: estabelecimento de uma base sólida para o desenvolvimento de *software*

  métodos: estratégias e técnicas para a construção de *software*

  ferramentas: suporte automatizado para processos e métodos

Modelos para processos

- Combinações e variações em torno de

  Análise: capturar informação sobre o domínio do problema e construir modelos operacionais para o sistema

  Projeto: transformar modelos da análise em modelos de elementos computacionais

  Codificação: implementar os elementos computacionais do sistema

  Teste: encontrar erros na implementação

  Manutenção: tudo de novo a cada mudança

- Resultado: código (o produto final)

Como desenvolver bom *software*?

- Balanço entre ênfase no produto *vs.* ênfase no processo
  - construções e linguagens de programação
  - estratégias e metodologias
- "Porém o produto não é o código, mas sim o conhecimento nele embutido" *P. Armour*
  - Encerrado o desenvolvimento, todo o *software* deveria ser reescrito
  - Problema das ordens de ignorância

As cinco ordens de ignorância

- OI-0: falta de ignorância
  - conhece alguma coisa e pode demonstrar esse conhecimento
- OI-1: falta de conhecimento
  - não sabe alguma coisa e sabe identificar este fato
- OI-2: falta de consciência
  - não sabe alguma coisa e nem sabe que não sabe
- OI-3: falta de processo
  - OI-2 e não sabe como fazer para descobrir que há coisas que não sabe
- OI-4: meta-ignorância
  - não sabe sobre as cinco ordens de ignorância

Ordens de ignorância e desenvolvimento de *software*

- OI-0: sistema funcionando corretamento
  - tem a resposta
- OI-1: variáveis são conhecidas
  - tem a questão
- OI-2: onde muitos projetos começam...
  - nem a resposta, nem a questão
- OI-3: onde mora o perigo...
  - metodologias de desenvolvimento devem mostrar onde falta conhecimento

O papel da orientação a objetos

- Desenvolvimento não começa na codificação
- Transição dos modelos de projeto para o código é facilitada pelo vocabulário comum
- Linguagens orientadas a objetos permitem expressar diretamente os conceitos usados no desenvolvimento orientado a objetos
  - classes, atributos e métodos
  - objetos
  - associações e composições
  - herança: reaproveitamento de definições
- "Com a orientação a objetos, você poderá reaproveitar código já desenvolvido e assim acelerar a produção de *software*."

Referências

1. Frederick P. Brooks, Jr. No Silver Bullet: Essence and accidents of software engineering. *IEEE Computer*, pp.10–19, April 1987. Disponível em `http://www.virtualschool.edu/mon/SoftwareEngineering/BrooksNoSilverBullet.html`.
2. Phillip G. Armour. The Five Orders of Ignorance. *Communications of the ACM* 43(10), pp.17–20, October 2000.