

Engenharia de Software Moderna

Cap. 1 - Introdução

Prof. Marco Tulio Valente

<https://engsoftmoderna.info>

Licença CC-BY; permite copiar, distribuir, adaptar etc; porém, **créditos devem ser dados ao autor dos slides**

Conferência da OTAN (Alemanha, 1968)

- 1a vez que o termo Engenharia de Software foi usado



Working Conference on **Software Engineering**

Comentário de participante da Conferência da OTAN

"Certos sistemas estão colocando demandas que estão além das nossas capacidades... **Estamos tendo dificuldades com grandes aplicações.**"

O que se estuda em ES?

1. Engenharia de Requisitos
2. Projeto de Software
3. Construção de Software
4. Testes de Software
5. Manutenção de Software
6. Gerência de Configuração
7. Gerência de Projetos



O que se estuda em ES? (cont.)

- 8. Processos de Software
- 9. Modelos de Software
- 10. Qualidade de Software
- 11. Prática Profissional
- 12. Aspectos Econômicos



Restante desta aula

- Vamos dar uma primeira visão de algumas dessas áreas
 - Objetivo: entendimento horizontal do que é ES
 - No resto do curso, vamos aprofundar nelas

Não Existe Bala de Prata



Frederick Brooks. No Silver Bullet - Essence and Accidents of Software Engineering. IEEE Computer, 1987.
Imagem de: https://twitter.com/zeljko_obren/status/909014656802574336

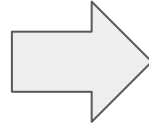
Motivo: Dificuldades Essenciais

Complexidade

Conformidade

Facilidade de Mudanças

Invisibilidade



Tornam Engenharia
de Software diferente
de outras engenharias

Vamos então comentar sobre algumas
áreas do SWEBOK

Requisitos de Software

- Requisitos: o que sistema deve fazer para atender aos seus clientes com qualidade de serviço

Requisitos Funcionais vs Não-Funcionais

- **Funcionais:** "o que" um sistema deve fazer
 - Funcionalidades ou serviços ele deve implementar
- **Não-funcionais:** "como" um sistema deve operar
 - Sob quais restrições e com qual qualidade de serviço

Exemplos de Requisitos Não-Funcionais

- Desempenho: dar o saldo da conta em 5 segundos
- Disponibilidade: estar no ar 99.99% do tempo
- Capacidade: armazenar dados de 1M de clientes
- Tolerância a falhas: continuar operando se São Paulo cair
- Segurança: criptografar dados trocados com as agências

Exemplos de Requisitos Não-Funcionais (cont.)

- Privacidade: não armazenar localização dos usuários
- Interoperabilidade: se integrar com os sistema do BACEN
- Manutenibilidade: bugs devem ser corrigidos em 24 hs
- Usabilidade: versão para celulares e tablets

1

*As Management
requested it.*



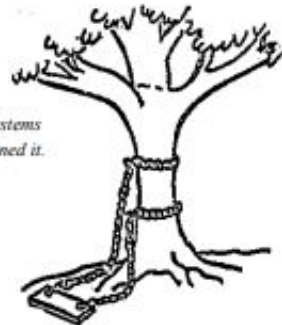
2

*As the Project Leader
defined it.*



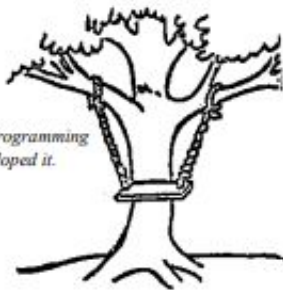
3

*As Systems
designed it.*



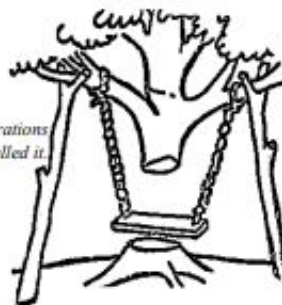
4

*As Programming
developed it.*



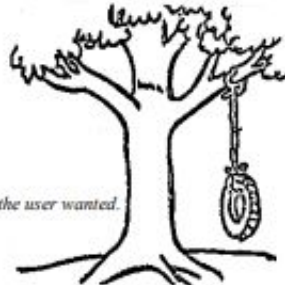
5

*As
Operations
installed it.*



6

What the user wanted.



Pre-1970 cartoon; origin unknown
Fonte: B. Meyer. Object Success, 1995.

Testes de Software

- Verificam se um programa apresenta um resultado esperado ao ser executado com casos de teste
- Podem ser:
 - Manuais
 - Automatizados (nosso foco)

Falha Famosa: Explosão do Ariane 5 (1996)



30 segundos depois

Custo do foguete e satélite:
US\$ 500 milhões

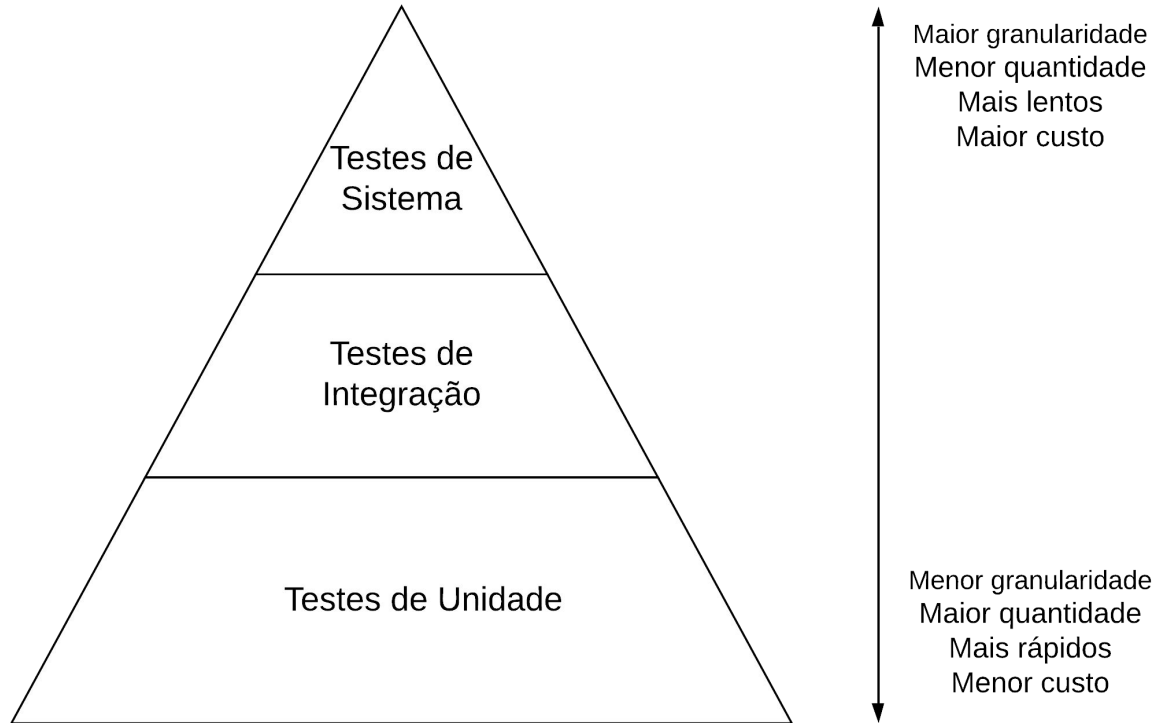


Photo of Ariane 501 Flight, a few seconds after explosion (Credits ESA 1996)

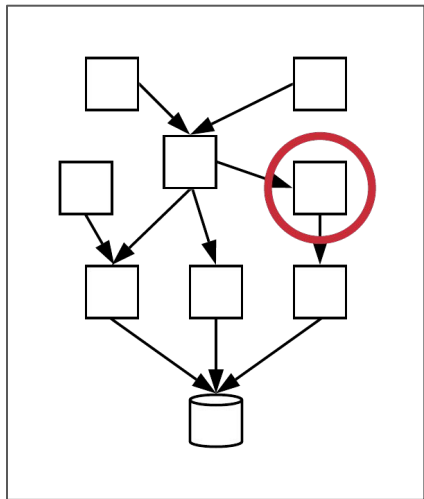
Relatório do Comitê de Investigação

- Explosão foi causada por uma falha de software
- Conversão de um real de 64 bits para um inteiro de 16 bits
- Como o real não "cabia" em 16 bits, a conversão falhou

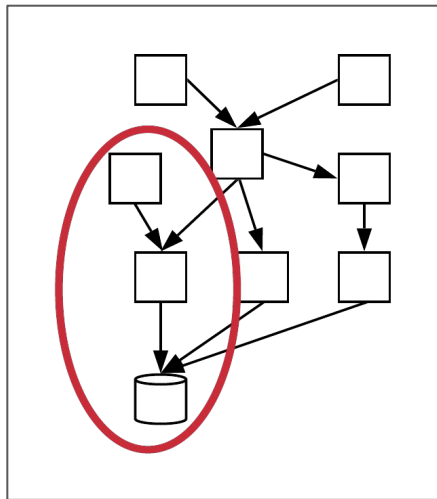
Pirâmide de Testes



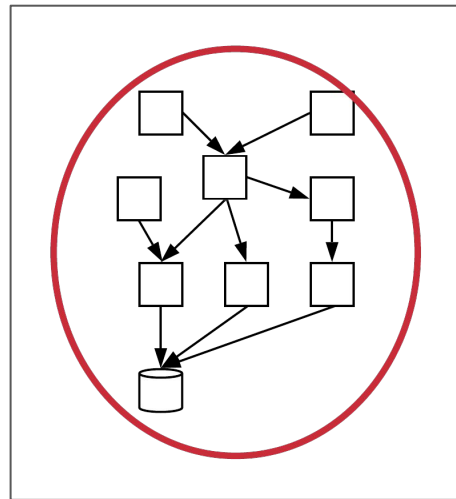
Tipos de Teste



Unidade



Integração



Sistema

Manutenção de Software

- Corretiva
- Preventiva
- Adaptativa
- Evolutiva
- Refactoring

Sistemas Legados

- Sistemas antigos, usando linguagens, SOs, BDs antigos
- Manutenção custosa e arriscada
- Muitas vezes, são importantes (legado \neq irrelevante)

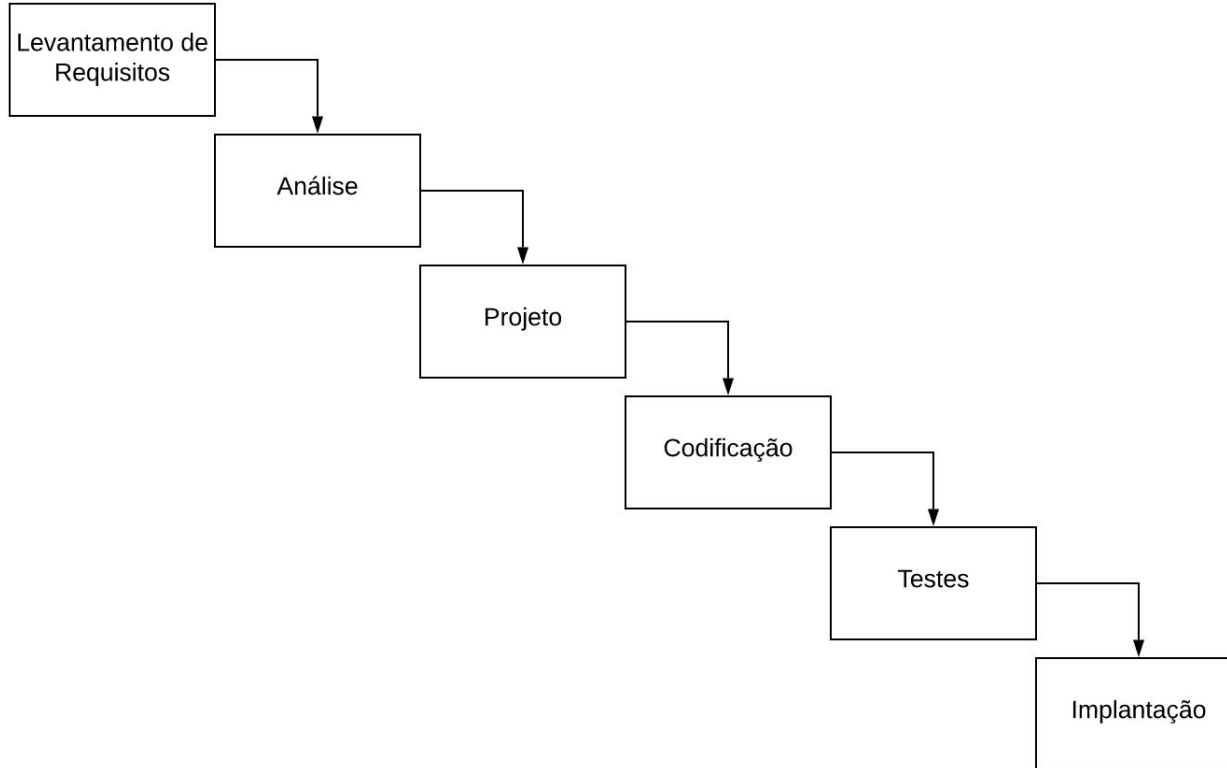
COBOL é muito comum em bancos

- Estima-se que existam ~200 bilhões de LOC em COBOL
- Maioria são sistemas de bancos
 - 95% das transações em ATMs são em COBOL
 - Um único banco europeu tem 250 MLOC em COBOL

Processos de Desenvolvimento de Software

- Processo de software: define as atividades que devem ser seguidas para construir um sistema de software
- Dois principais modelos:
 - Waterfall ("cascata")
 - Ágil

Modelo em Cascata

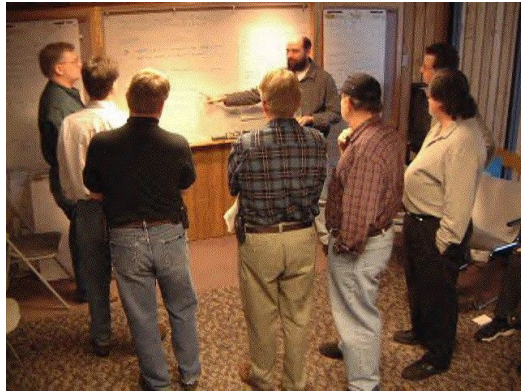


Problemas com Modelo Waterfall

- Requisitos mudam com frequência
 - Levantamento completo de requisitos demanda tempo
 - Quando ficar pronto, o "mundo já mudou"
 - Clientes às vezes não sabem o que querem
- Documentações de software são verbosas
 - E rapidamente se tornam obsoletas

Manifesto Ágil (2001)

- Encontro de 17 engenheiros de software em Utah
- Crítica a modelos sequenciais e pesados
- Novo modelo: incremental e iterativo



Profundo impacto na indústria de software



Março 2019



Mai 2020

Aspectos Éticos

- Engenheiros de Software começam a questionar o uso que as empresas fazem do software desenvolvido por eles

Cybersecurity

Google Engineers Refused to Build Security Tool to Win Military Contracts

A work boycott from the Group of Nine is yet another hurdle to the company's efforts to compete for sensitive government work.

<https://www.bloomberg.com/news/articles/2018-06-21/google-engineers-refused-to-build-security-tool-to-win-military-contracts>

Para finalizar: Tipos ABC de sistemas

Tipos ABC de sistemas

- Classificação proposta por Bertrand Meyer
- Três tipos de software:
 - Sistemas C (Casuais)
 - Sistemas B (Business)
 - Sistemas A (Acute)

Sistemas C (Casuais)

- Tipo muito comum de sistema
- Sistemas pequenos, sem muita importância
- Podem ter bugs; às vezes, são descartáveis
- Desenvolvidos por 1-2 engenheiros
- **Não se beneficiam tanto do que vamos estudar**
- Risco: "over-engineering"

Sistemas B (Business)

- Sistemas importantes para uma organização
- **Sistemas que se beneficiam do que veremos no curso**
- Risco: não usarem técnicas de ES e se tornarem um passivo, em vez de um ativo para as organizações

Sistemas A (Acute)

- Sistemas onde nada pode dar errado, pois o custo é imenso, em termos de vidas humanas e/ou \$\$\$
- Sistemas de missão crítica



Metrô



Aviação



Medicina

Sistemas A (Acute)

- Requerem certificações
- **Fora do escopo do nosso curso**

Document Title

DO-178C - Software Considerations in Airborne Systems and Equipment Certification

Description

This document provides recommendations for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. Compliance with the objectives of DO-178C is the primary means of obtaining approval of software used in civil aviation products.

Document Number

DO-178C

Format

Hard Copy

Committee

SC-205

Issue Date

12/13/2011

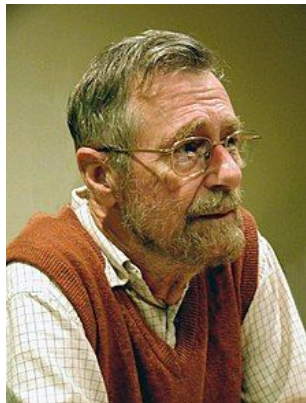
Exercícios

1. Os custos com manutenção podem alcançar 80% dos custos totais alocados a um projeto de software durante o seu ciclo de vida. Por que esse valor é tão alto?
2. Suponha que você tenha que construir uma ponte. Como seria o projeto dessa ponte usando:
 - a. Um método waterfall
 - b. Um método ágil
3. Refactoring é uma transformação de código que preserva comportamento. Qual o significado da expressão preservar comportamento?

4. Em testes, existe uma frase muito famosa, de autoria de Edsger W. Dijkstra, que diz que:

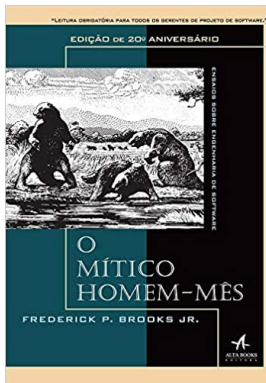
“testes de software mostram a presença de bugs, mas não a sua ausência.”

Por que testes são incapazes de mostrar a ausência de bugs?

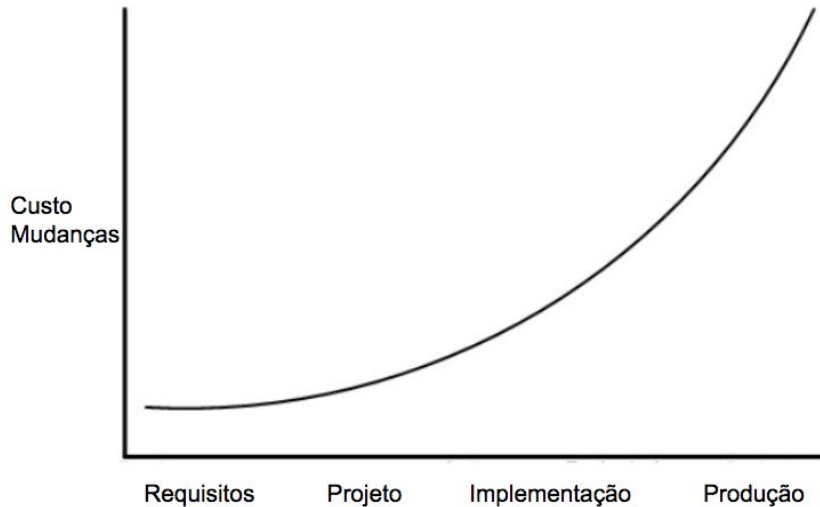


5. Em gerência de projetos de software, existe uma lei empírica muito famosa, chamada **Lei de Brooks**, que diz que
- “incluir novos devs em um projeto que está atrasado, vai deixá-lo mais atrasado ainda.”**

Por que essa lei tende a ser verdadeira?



6. Seja o seguinte gráfico, que mostra — para um sistema — como os custos de mudanças variam conforme a fase do desenvolvimento. Qual método de desenvolvimento você recomendaria para esse sistema?



7. Em 2015, descobriu-se que o software instalado em milhões de carros da Volkswagen comportava-se de forma diferente quando testado em um laboratório de certificação. Nessas situações, o carro emitia poluentes dentro das normas. Fora do laboratório, ele emitia mais poluentes... Ou seja, o código incluía um “if” como o seguinte (meramente ilustrativo). O que você faria se seu chefe pedisse para escrever um if como esse?

```
if "Carro sendo testado em um laboratório"  
    "Emita poluentes dentro das normas"  
else  
    "Emita poluentes fora das normas"
```

Marque V ou F

- (a) Complexidade, conformidade, portabilidade e invisibilidade são dificuldades essenciais em Engenharia de Software.
- (b) São exemplos de requisitos não-funcionais em um aplicativo de banco: usabilidade, segurança, desempenho e suporte ao PIX.
- (c) Uma empresa X decidiu terceirizar o desenvolvimento de um sistema com uma empresa Y. Então, como é uma empresa externa, Y não é um dos stakeholders desse sistema.
- (d) São exemplos de refactorings: extrair um método menor (a partir de um método maior); renomear uma variável local; mover um método para uma classe mais apropriada.

Fim