

# Engenharia de Software

Prof. Alan Braz  
[alanbraz@gmail.com](mailto:alanbraz@gmail.com)

***"Software: Esses programas dão instruções para a CPU, que processa bilhões de fatos minúsculos chamados bytes, e dentro de uma fração de segundo que lhe envia uma mensagem de erro que requer que você ligue para o suporte ao cliente on-line que te coloca em espera durante o tempo de vida de uma rena."***



***Dave Barry  
(comediante americano)***

# Alinhando Expectativas



## •Apresentações:

- Nome
- Empresa
- Tempo de Experiência
- Formação Acadêmica
- Onde mora
- Expectativas (pós)

# Apresentação



## •Alan Braz

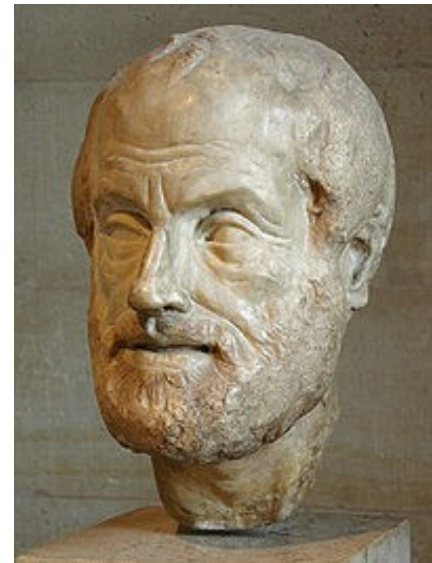
- Formação:
  - Graduado em Ciência de Computação – UNICAMP
  - Mestrando em Eng. Soft. Métodos Ágeis – UNICAMP
- Empresa: IBM (desde 2005)
- Experiências: Administração de Empresas, Desenvolvedor e Arquiteto JavaEE, Instrutor Java, JavaEE e Agile.
  - Professor de Eng. Soft, RUP e Métodos Ágeis desde 2009 no curso de ES-SOA
- Projetos Atuais na IBM:
  - Arquiteto JavaEE
  - Consultor e Professor de Agile
- Contato: alanbraz@gmail.com

# Objetivos

- Ao final desta disciplina os alunos estarão aptos a:
  - Compreender, com uma **visão abrangente e gerencial**, os conceitos relacionados aos temas da Engenharia de Software
  - Relacionar os **conceitos, princípios e técnicas** de Engenharia de Software aos de **Gestão de Projetos** e evidenciar suas aplicações
  - Decidir quais as melhores práticas de **Engenharia de Software** a serem aplicadas aos seus Projetos de TI

***"As raízes da educação são amargas,  
mas o fruto é doce."***

***Aristóteles***



# Bibliografia Básica



- PRESSMAN, R. S. Software Engineering : A Practitioner's Approach – Fifth Edition. Editora McGraw-Hill, 2001.
- SOMMERVILLE, I. Engenharia de Software, 8a. Edição. Editora Addison Wesley, 2007.
- CÔRTEZ, M. L.; CHIOSSI, T. C. Modelos de Qualidade de Software. Campinas : Editora da Unicamp, 2001.
- FERNANDES, A. A.; TEIXEIRA, D. S. Fábrica de Software – Implantação e Gestão de Operações. São Paulo : Editora Atlas, 2004.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). NBR ISO 9001:2000 – Sistema de Gestão da Qualidade - Requisitos. Rio de Janeiro : ABNT, 2001.
- PAULK, M. C. et al. Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-024. Pittsburgh, PA : Software Engineering Institute, 1993.
- PAULK, M. C. et al. The Capability Maturity Model : Guidelines for Improving the Software Processes. Indianapolis, IN : Addison-Wesley, 2003.
- THE INSTITUTE do ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE). SWEBOOK – Guide to the Software Engineering Body do Knowledge : 2004 Version. Los Alamitos, CA : IEEE Computer Society, 2004.
- CMMI PRODUCT TEAM (CPT). Capability Maturity Model Integration (CMMI), Version 1.1, Staged Representation, CMU/SEI-2002-TR-012. Pittsburgh, PA : Software Engineering Institute, 2002.
- DINSMORE, P. C. et al. Como se tornar um profissional em Gerenciamento de Projetos. Rio de Janeiro : Qualitymark Editora, 2004.

# Bibliografia Recomendada

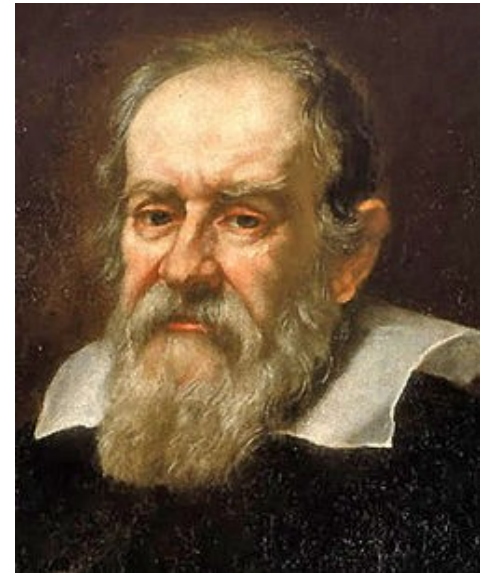


- AAEN, I.; BOTCHER, P.; MATHIASSEN, L. The Software Factory : Contributions and Illusions. In : PROCEEDINGS do THE TWENTIETH INFORMATION SYSTEMS RESEARCH SEMINAR IN SCANDINAVIA, 1997, Oslo.
- ALVES, E. P.; FALBO, R. A. Implantando um Programa de Melhoria de Processo : Uma Experiência Prática. In : WORKSHOP DE QUALIDADE DE SOFTWARE, 8., 2001, Rio de Janeiro. Anais..., Brasília : Ministério de Ciência e Tecnologia, 2001, p. 16-23.
- BRUEGGE, B. Requirements Analysis and Modelling. IEEE Software, p. 361-364, 1995.
- BUHNE, S. et al. Defining Requirements at Different Levels do Abstraction. IEEE Software, Proceedings do the 12th IEEE International Requirements Engineering Conference, 2004.
- CANTONE, G. Software Factory : Modeling the Improvement. Italy : University do Rome "Tor Vergata", p. 124-129, s.ed., s.d.
- CARVALHO, A. E. S. et al. Uma Estratégia para Implantação de uma Gerência de Requisitos visando a Melhoria dos Processos de Software. Recife : Universidade Federal de Pernambuco, 2001.
- FERNSTROM, C. et al. Software Factory Principles, Architecture, and Experiments. IEEE Software, Vol. 9, No. 2, p. 36-44, march/april 1992.
- MELLO, C. H. P. et al. ISO 9001:2000 – Sistema de Gestão da Qualidade para Operações de Produção e Serviços. São Paulo : Editora Atlas, 2002.
- WIEGERS, K. E. Software Requirements : Practical Techniques for gathering and managing requirements throughout the product development cycle. Editora Microsoft Press, 1999.
- FOWLER, M. UML Essencial – Um brece guia para a linguagem-padrão de modelagem de objetos. Editora Bookman, 2005.



***"Você não pode ensinar nada a um homem,  
você pode apenas ajudá-lo a encontrá-lo  
dentro de si mesmo."***

***Galileu Galilei***



# Dinâmica das Aulas

- Além das técnicas tradicionais de ensino-aprendizagem, serão utilizadas técnicas tais como:

- Workshops
- Trabalhos em grupo
- Leituras de artigo
- Seminários



# Plano de Aula – 1º Dia



## •Software

- Definição
- Classificações
- Disciplinas
- História & Tendências
- Crise do Software

## •Engenharia de Software

- Conceitos
- Ciclo de Vida: Processos, métodos e ferramentas

# Plano de Aula – 2º Dia

## • Modelos de Processo de Software

- Cascata
- Espiral
- RUP
- Ágil (XP e Scrum)

## • Workshop sobre Ciclos de Vida de Projeto

## • Fábricas de Software

- Conceitos



# Plano de Aula – 3º Dia

## •Requisitos

- Conceitos
- Tipos de requisitos
- Atividades de gestão e engenharia
- Técnicas de elicitação
- Fontes de requisitos

## •Estimativas

- Conceitos
- Estimativas de Tamanho e Esforço : FPA, UCP, LOC e WBS

## •Métricas de Software

- Conceitos
- Possíveis Indicadores

# Plano de Aula – 4º Dia



- Relacionamento entre PMBOK e Engenharia de Software
- Melhoria de Processos e Modelos de Qualidade
  - ISO9001:2000
  - CMMI
  - MPSBR
- Introdução ao Scrum

# Avaliação

- **Serão realizadas 2 (duas) avaliações com base em um estudo de caso:**
- **Avaliação SWOT e decisão sobre ciclo de vida**
  - A ser executado na 2ª. Aula, em classe
- **Documento de Requisitos**
  - A ser elaborado extra-classe e apresentado e discutido na 3ª. Aula

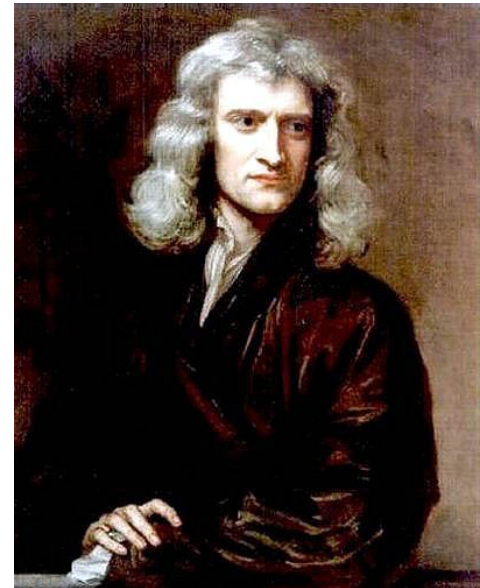
# Software

Definição  
Classificações  
Disciplinas  
História  
Tendências



***“Se eu fui capaz de ver mais longe do que outros, foi porque eu estava sobre os ombros de gigantes.”***

***Sir Isaac Newton***



# Definição

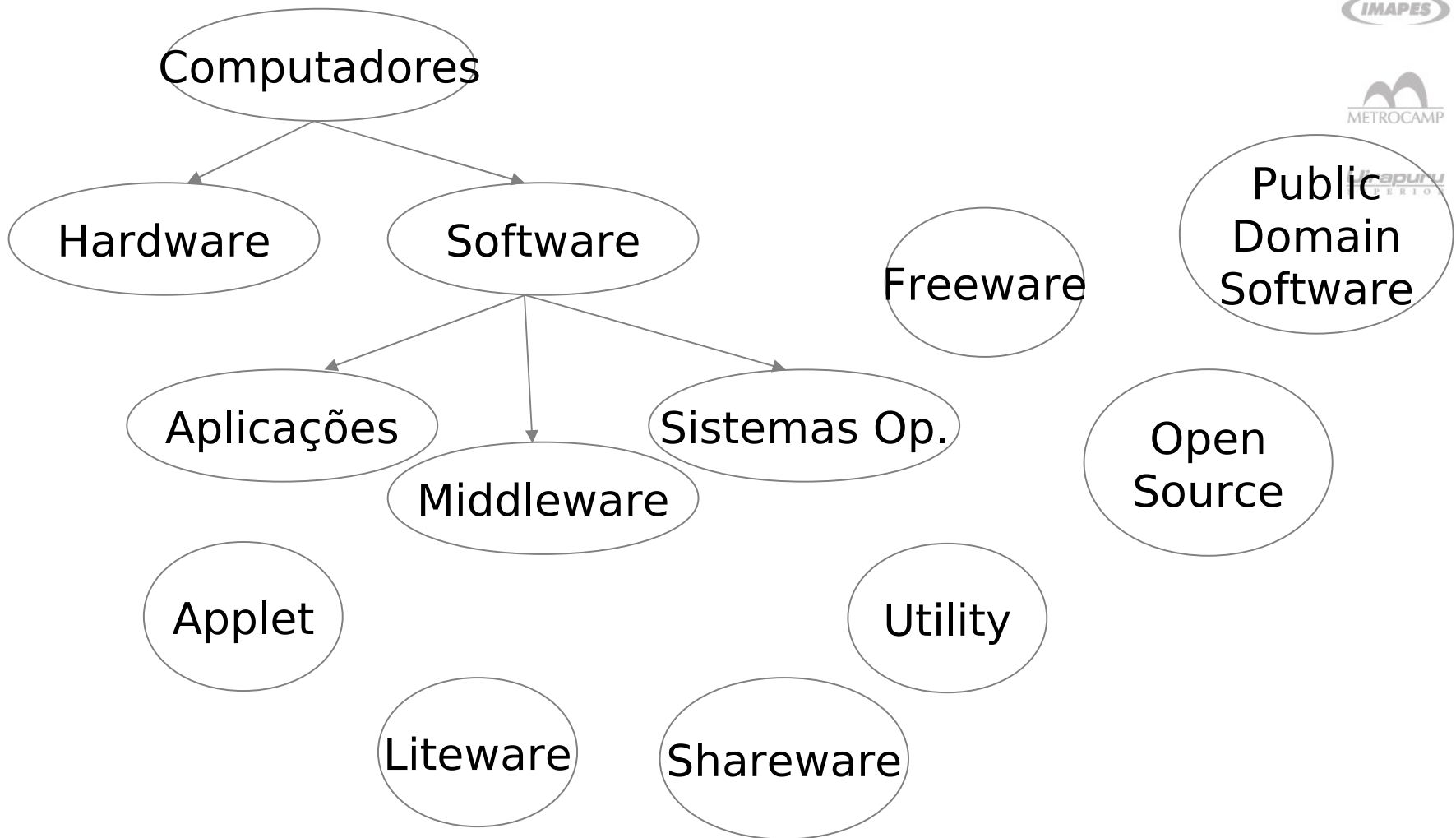
Termo Geral para vários tipos de programas usados para operar computadores e dispositivos afins.

*Fonte: SearchSOA.com Definitions (Powered by WhatIs.com)*  
*<http://searchsoa.techtarget.com/definition/software>*

Sequência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado/informação.

*Fonte: [pt.wikipedia.org/wiki/Software](http://pt.wikipedia.org/wiki/Software)*

# Taxonomia



## 3 grandes classes:

- ✓ ***Software de Sistema;***
- ✓ ***Software de Programação;***
- ✓ ***Software Aplicativo.***

Sistemas operacionais,  
Servidores,  
Utilitários,  
Device drivers,  
Windowing systems

Compiladores,  
Debuggers,  
Interpretadores,  
Linkers,  
Text editors.  
(IDE)

Automação Industrial,  
Business software (ERP),  
Games,  
Telecomm (internet),  
Databases Etc.

# Aplicações

## Pressman, divisão em 7 classes:

- ✓ **Básico;**
- ✓ **Tempo Real;**
- ✓ **Comercial;**
- ✓ **Científico / Engenharia;**
- ✓ **Embutido;**
- ✓ **Computador Pessoal;**
- ✓ **Inteligência Artificial.**

Programas de apoio a outros programas

Monitora, analisa e controla eventos do mundo real

Operações comerciais e tomadas de decisões administrativas

Algoritmos de processamento de números

Controla produtos e sistemas de mercados industriais e de consumo

Processamento de textos, planilhas eletrônicas, diversões etc.

Algoritmos não numéricos para resolver problemas complexos

## Tópicos:

- ✓ **Arquitetura;**
- ✓ **Codificação**
- ✓ **Documentação;**
- ✓ **Biblioteca;**
- ✓ **Execução;**
- ✓ **Qualidade e confiabilidade;**
- ✓ **Padrão;**
- ✓ **Licenciamento;**
- ✓ **Ética.**

# Conceito

## • Programação – Conceito

- Mitologia grega → Hefesto → servos mecânicos

## • História

- Máquina de Anticítera → ~ 1 séc. a.C.
- Al-Jazari (1136 – 1206) → livro com 50 dispositivos mecânicos
- Joseph Marie Jacquard (1752 – 1834) → Tear de Jacquard (1801)
- Charles Babbage (1791 – 1871) → máquina analítica (1830)
  - Augusta Ada King, Condessa de Lovelace (1815 – 1852)
- Herman Hollerith (1860 – 1929) → tabulador a cartão perfurado
  - Em 1896 fundou a [Tabulating Machine Company](#)
  - Em 1911 fundiu-se a outras 3 constituindo a [Computing Tabulating Recording Corporation \(CTR\)](#)
  - Em 1924 foi renomeada para [IBM](#) - Thomas J. Watson

# Máquina de Anticítera



Descoberto em 1901

Calcula movimentos  
astronômicos

Data estimada do  
naufrágio é 65 a.C.

Mais valioso que a  
Mona Lisa

*Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism* - . Freeth, Y.  
Bitsakis, X. Moussas, et al – Nature - 30 November 2006

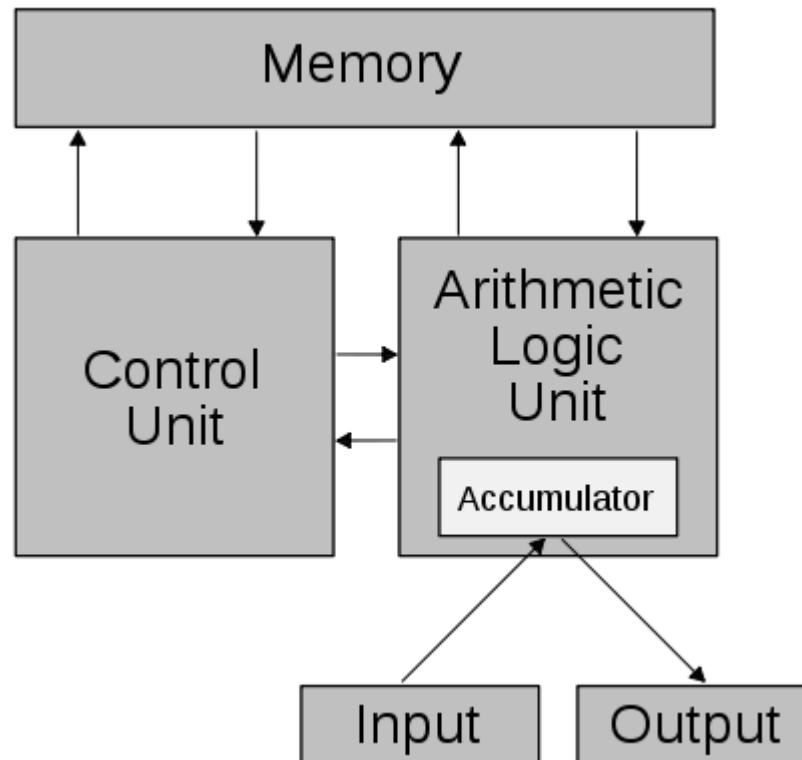


# A Invenção do Software

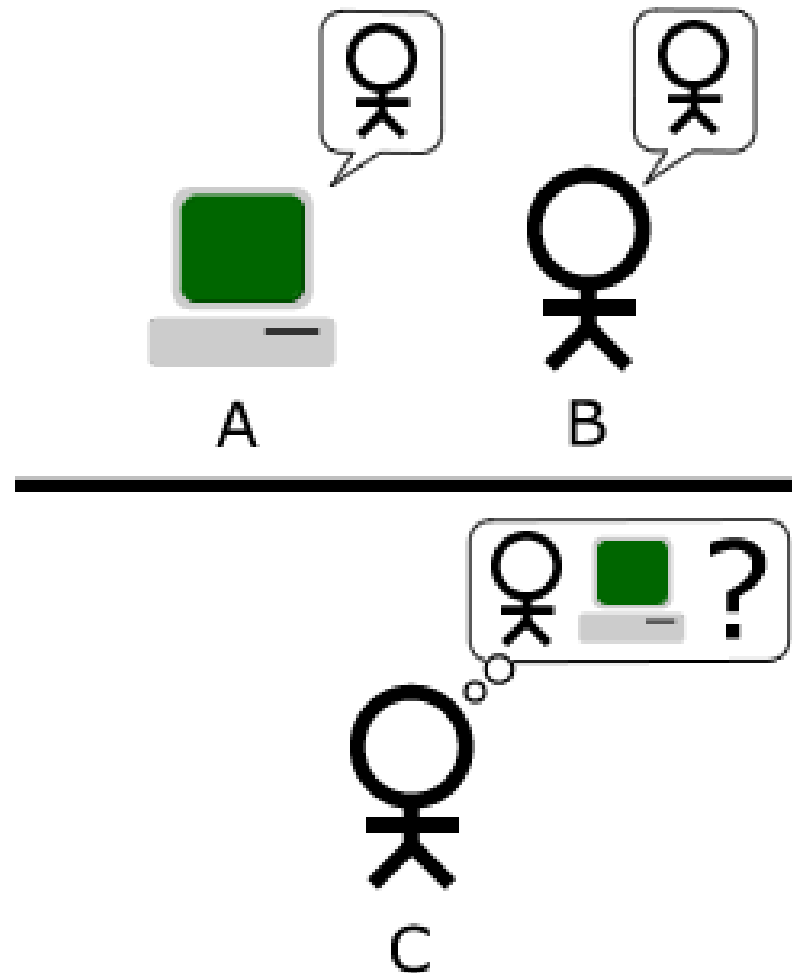
## •Um pouco mais de História

- John von Neumann (1903 – 1957) → The von Neumann architecture
  - Programas armazenados na memória
  - Evolução dos programas em “program-controlled computers”
    - Colossus e ENIAC
    - Configuração de Switches e “patch leads” para roteamento dos dados entre as unidades funcionais
- Alan Mathison Turing (1912 – 1954) → conceitos de algoritmo e computação com a Máquina de Turing (1937) e o Teste de Turing (1950)
- Anos 40
  - Primeiros computadores
  - Assembly Language
  - Primeiras linguagens (ENIAC coding system, Plankalkül, C-10)

# Von Neumann architecture



# Turing Test



# A Inovação

## •Anos 50 e 60

- O hardware sofreu contínuas mudanças
- O software era uma arte "secundária" para a qual havia poucos métodos sistemáticos
- O hardware era de propósito geral
- O software era específico para cada aplicação
- Não havia documentação

# A Inovação

## •Anos 50 e 60

- As 3 Primeiras linguagens modernas
  - FORTRAN (55), **FOR**mula **TRAN**slator, John W. Backus et al.;
  - LISP, **LIS**t **P**rocessor, John McCarthy et al.;
  - COBOL, **CO**mmon **B**usiness **O**riented **L**anguage, Short Range Committee (Grace Hopper)
- Algol 60 Report (**ALGO**rithmic **L**anguage) - 1963
  - Estrutura de blocos
  - Escopo léxico
  - Backus–Naur Form
    - Metasintaxe para qualquer linguagem de programação
    - 2 conjuntos de regras: léxicas e sintáticas

# Projeto e Implementação



- 1965 - 1975
  - Multiprogramação e sistemas multiusuários
  - Técnicas interativas
  - Sistemas de tempo real
  - 1a geração de SGBD's
  - Produto de software - software houses
  - Bibliotecas de Software
  - Manutenção quase impossível

# Projeto e Implementação

## •1967-1978: Paradigmas fundamentais

- Florescimento intensivo das linguagens de programação
- Simula 67 – 1ª a suportar orientação a objeto (Norwegian Computing Center)
- C - Dennis Ritchie e Ken Thompson, Bell Labs, entre 1969 e 1973
- Smalltalk (~1970) – projeto completo de linguagem OO
- Prolog, 1972
- ML, Robin Milner (1973) → polimorfismo
- Programação estruturada → morte do GOTO
- Importantes linguagens desse período:
  - 1970 - Pascal
  - 1970 - Forth
  - 1972 - C
  - 1972 - Smalltalk
  - 1972 - Prolog
  - 1973 - ML
  - 1978 - SQL

# Projeto e Implementação

- 1975 – final dos anos 90
  - Sistemas distribuídos
  - Redes locais e globais
  - Uso generalizado de microprocessadores - produtos inteligentes
  - Hardware de baixo custo
  - Impacto de consumo



# Projeto e Implementação

- Anos 80: consolidação, modularização, performance

- Sem novos paradigmas
- Evolução das idéias das linguagens existentes
- RISC – Reduced Instruction Set Computer
  - Representa uma estratégia para concepção de CPUs enfatizando a visão de que instruções simplificadas podem prover melhor desempenho
- Importantes linguagens desse período
  - 1983 - Ada
  - 1983 - C++
  - 1985 - Eiffel
  - 1987 - Perl
  - 1989 - FL (Backus)

# Projeto e Implementação



## •Anos 90: a era da Internet

- Sem novidades fundamentais, mas muita combinação e amadurecimento das “velhas” idéias
- Grande concentração de esforços na produtividade do programador
- Surgimento das linguagens “Rapid Application Development”
- Scripting languages – sistemas web based
- Importantes linguagens desse período
  - 1990 - Haskell
  - 1991 - Python
  - 1991 - Java
  - 1993 - Ruby
  - 1993 - Lua
  - 1994 - ANSI Common Lisp
  - 1995 - JavaScript
  - 1995 - PHP
  - 2000 - C#
  - 2008 - JavaFX Script

# O Futuro



## •Tendências atuais

- Orientação a aspectos
  - Aspect-oriented programming (AOP)
  - Desenvolvimento de software orientado a componentização
- Desenvolvimento ágil
  - SCRUM, XP, Lean software development
- Projeto orientado a modelo
- Linha de produtos de software
  - Metodologia sistemática de produção de famílias de software
    - Reuso
    - Fabricação
- Ênfase crescente em distribuição e mobilidade.
- Integração de bases de dados (XML)
- Orientação à Serviços

# Evolução da Tecnologia da Informação

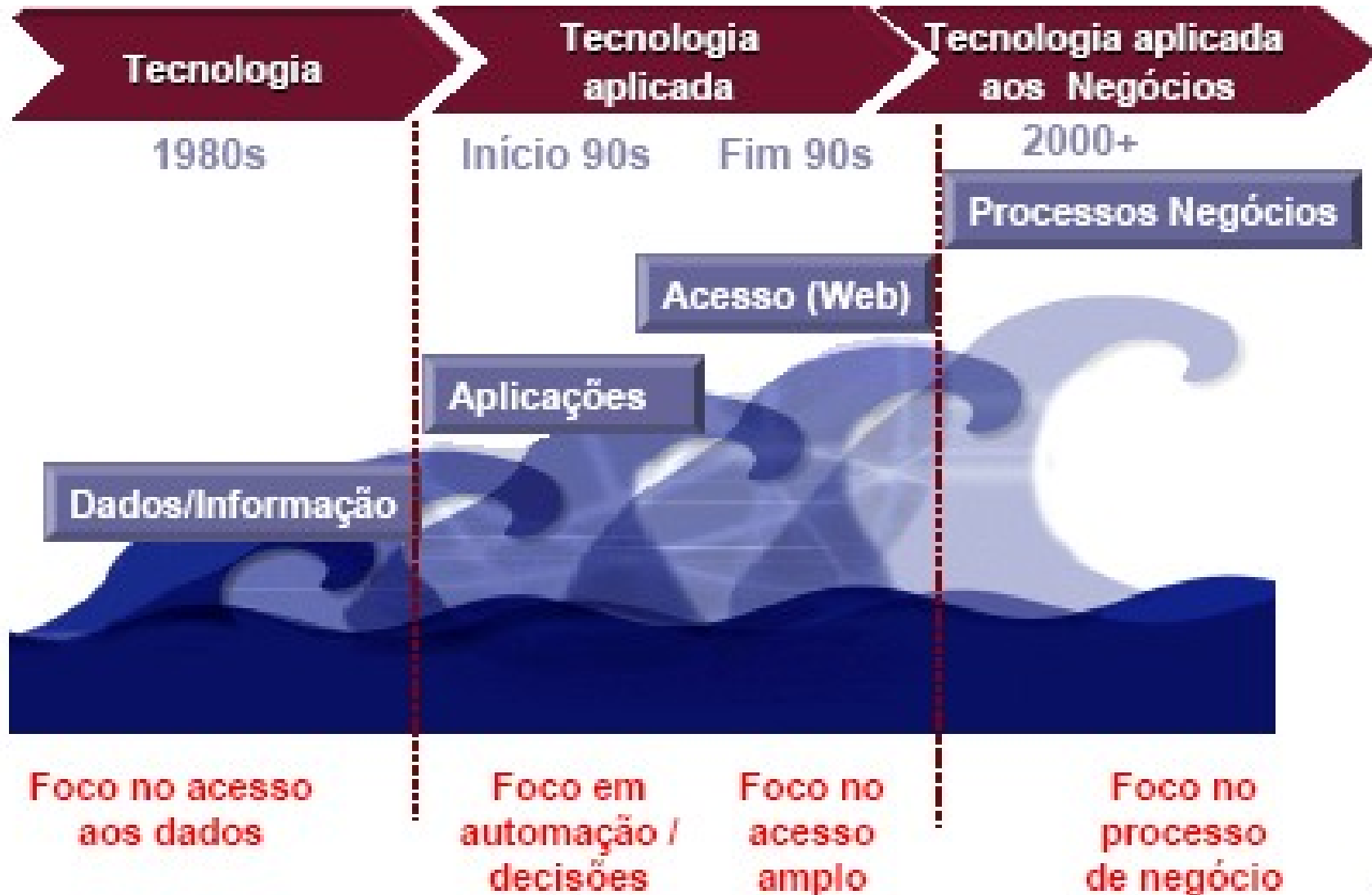


IBTA

MAPES

TROCAMP

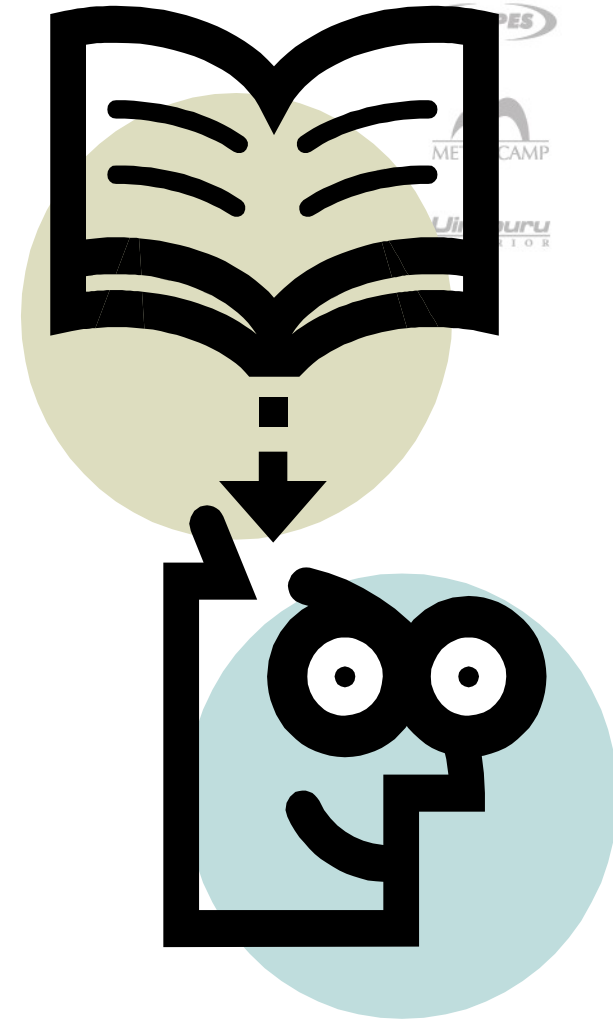
rapuru  
PERIOR



## Perguntas :

**Em sua opinião, por que é sempre tão difícil criar e dar manutenção em um software?**

**E quais são as “verdades” e “mentiras” que se tornaram mitos na hora de implementar um software?**



# Mitos do Software

- **Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?**

## **Realidade:**

**Será que o manual é usado?**

**Os profissionais sabem que ele existe?**

**Ele reflete a prática moderna de desenvolvimento de software?**

**Ele é completo?**

# Mitos do Software

- **Meu pessoal tem ferramentas de desenvolvimento de software de última geração, afinal lhes compramos os mais novos computadores e CASEs.**

**Realidade:**

**É preciso muito mais do que os mais recentes computadores ou CASEs para se fazer um desenvolvimento de software de alta qualidade.**

- **Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso.**

**Realidade:**

**O desenvolvimento de software não é um processo mecânico igual à manufatura.**

**Acrescentar pessoas em um projeto torna-o ainda mais atrasado. Pessoas podem ser acrescentadas, mas somente de uma forma planejada.**



- **Uma declaração geral dos objetivos é suficiente para se começar a escrever programas - podemos preencher os detalhes mais tarde.**

**Realidade:**

**Uma definição inicial ruim é a principal causa de fracassos dos esforços de desenvolvimento de software.**

**É fundamental uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.**

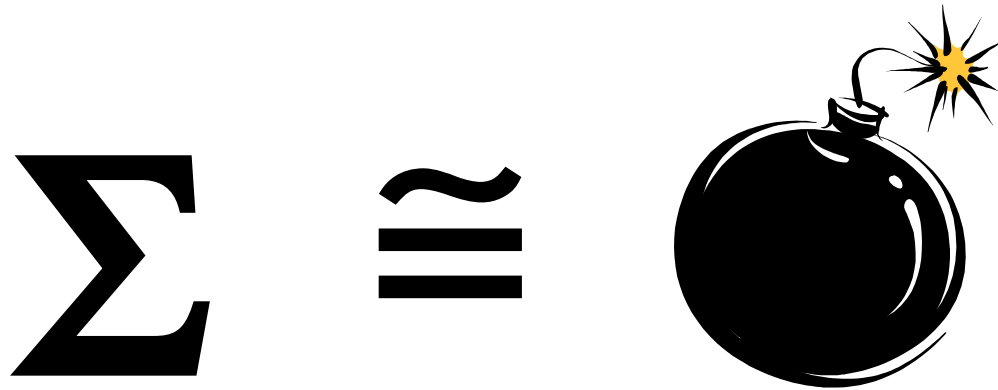
- **Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.**

**Realidade:**

**Uma mudança, quando solicitada tardiamente num projeto, pode ser muito mais dispendiosa do que a mesma mudança solicitada nas fases iniciais.**

# Crise do Software (1972)

- Estimativas de **prazo** e de **custo** ↑
- **Produtividade** das pessoas ↓
- **Qualidade** de software ↓
- Software **difícil de manter** ↑



# Therac-25



- Equipamento de Radioterapia.
- Entre 1985 e 1987 se envolveu em 6 acidentes, causando 5 mortes por overdoses de radiação.
- Software foi adaptado de uma antecessora, Therac-6:
  - O código não havia sido revisado/testado independentemente;
  - A documentação do sistema fornecida aos usuários não explicava o significado dos códigos de erro que a máquina retornava
  - A primeira reação dos funcionários da AECL (fabricante da máquina) foi negar a existência de erros.

# Denver International Airport



- **Custo do projeto: US\$ 4.9 bilhões**
  - 100 mil passageiros por dia
  - 1,200 vôos
  - 53 milhas quadradas
  - 94 portões de embarque e desembarque
  - 6 pistas de pouso / decolagem

# Denver International Airport



- Erros no sistema automático de transporte de bagagens (*misloaded, misrouted, jammed*):
  - Atraso na abertura do aeroporto com custo total estimado em US\$360 Milhões
- US\$86 milhões para consertar o sistema

# Ariane 5

veris

IBTA

IMAPES

METROCAMP

Uirapuru  
SUPERIOR



# Ariane 5



- Projeto da Agência Espacial Européia
  - 10 anos
  - US\$ 8 Bilhões
- Capacidade 6 toneladas.
- Garante supremacia europeia no espaço.



# Vôo inaugural em 4/junho/1996

veris

IBTA

IMAPES

METROCAMP

Uirapuru  
SUPERIOR



# Resultado



- Explosão 40 segundos após a decolagem.
- Destruição do foguete e carga avaliada em US\$500 milhões.

# Causa...

- Erro na conversão de um número  
Float de 64bits para um Inteiro de 32bits
- Ironia...
  - O resultado desta conversão não era mais necessário após a decolagem...



The rocket exploded seconds after launching

# O que é Software?

## ■ Definição - Software é:

- 1º - **instruções** (programas de computador) que, quando executadas, produzem a função e o desempenho desejados;
- 2º - estruturas de **dados** que permitem a manipulação das informações;
- 3º - **documentos** que descrevem a operação e uso dos programas.

# Características do Software - 1



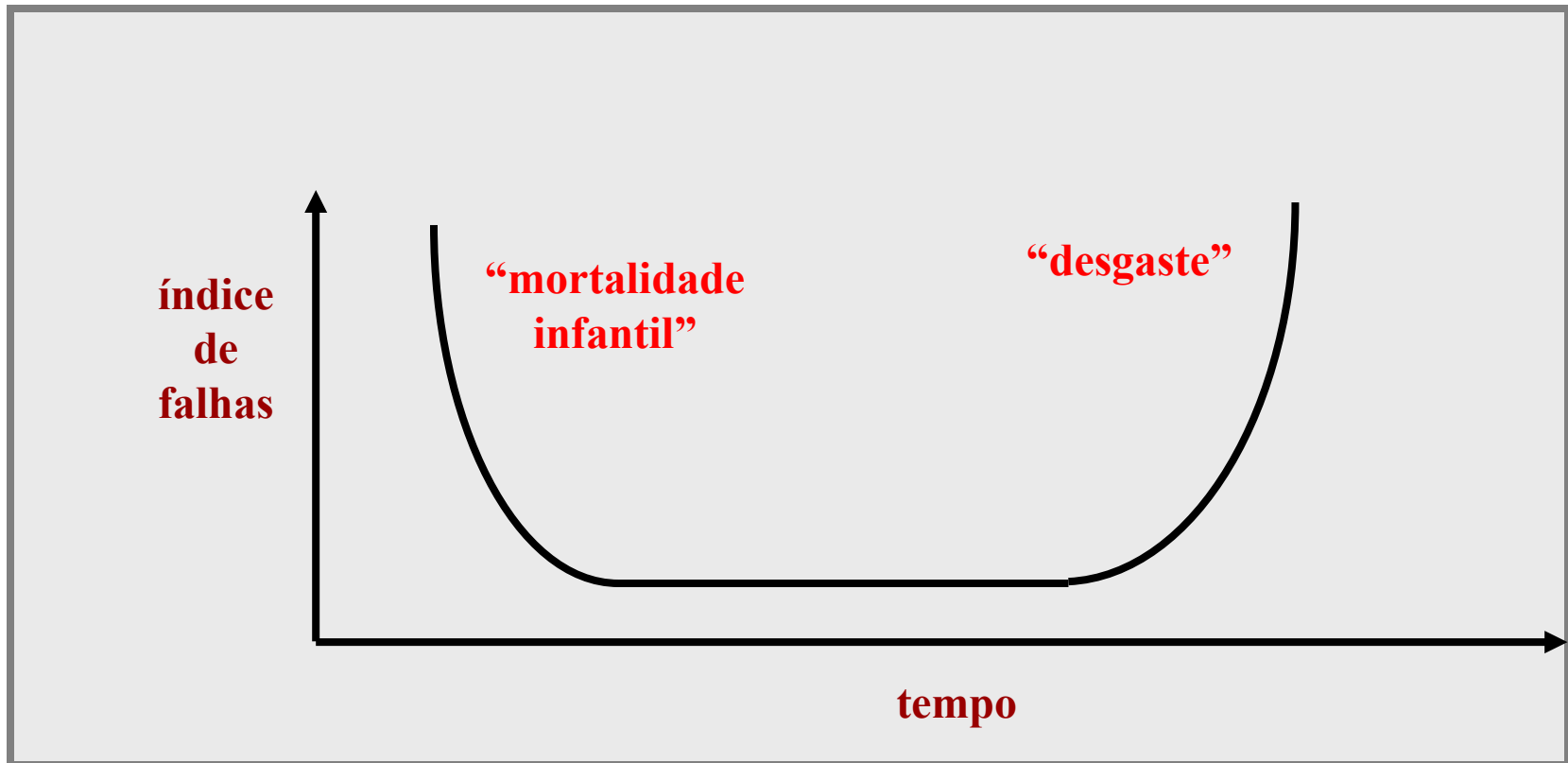
- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico:
  - Custos são concentrados no trabalho de engenharia.
  - Projetos não podem ser geridos como projetos de manufatura.
  - “Fábrica de Software!”

# Características do Software - 2

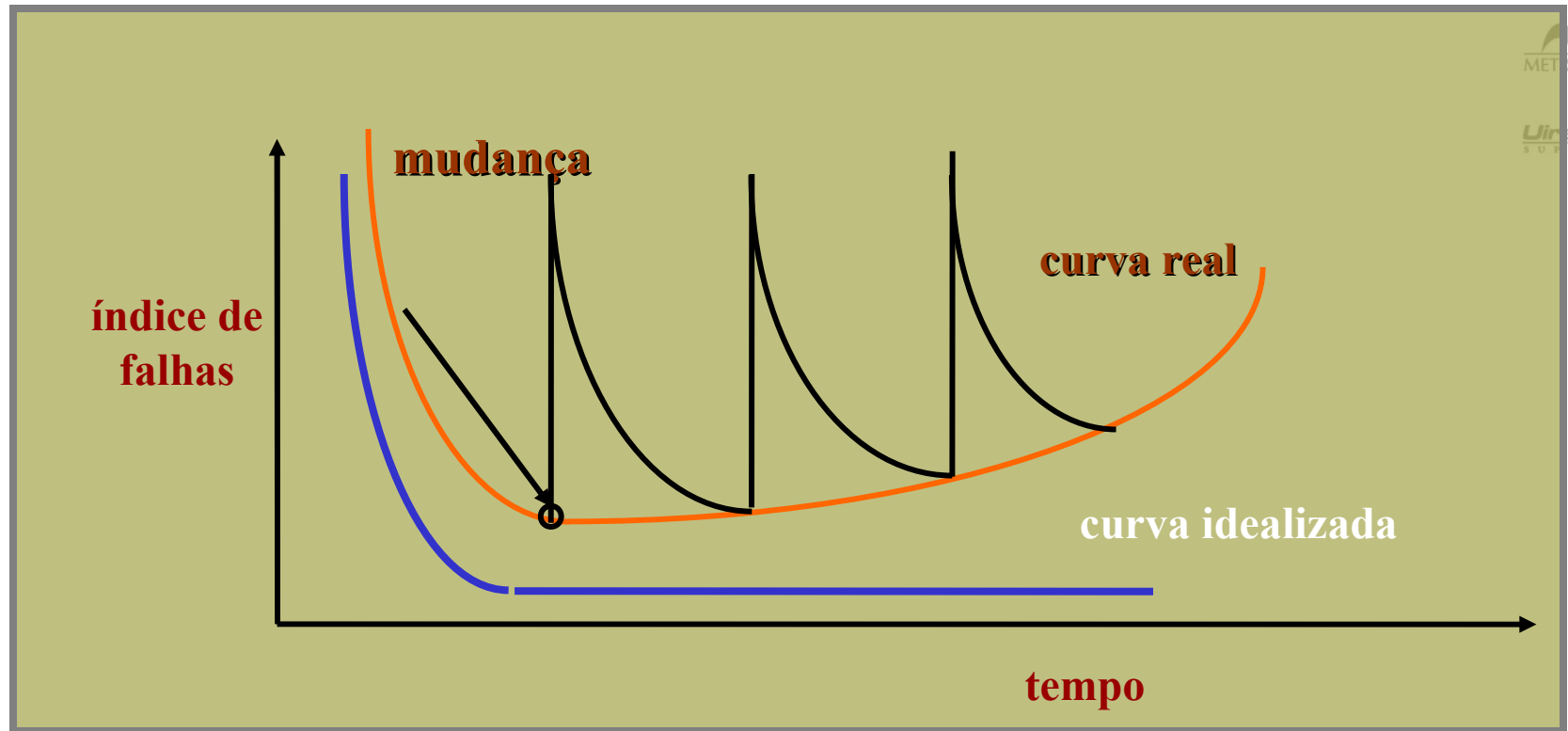


- **Software não desgasta!**
  - Software não é sensível aos problemas ambientais que fazem com que o hardware se desgaste.
  - Toda falha indica erro de projeto ou implementação
  - Manutenção do SW é mais complicada que a do HW.

# Curva de falhas para o Hardware



# Curva de falhas do Software





# Características do Software - 3



- A maioria dos softwares é feita sob medida e não montada a partir de componentes existentes.
- != Hardware.
- Situação esta mudando:
  - Orientação a objetos.
  - Reusabilidade  
(diminui custos e melhora projetos).

# Quais são os problemas?

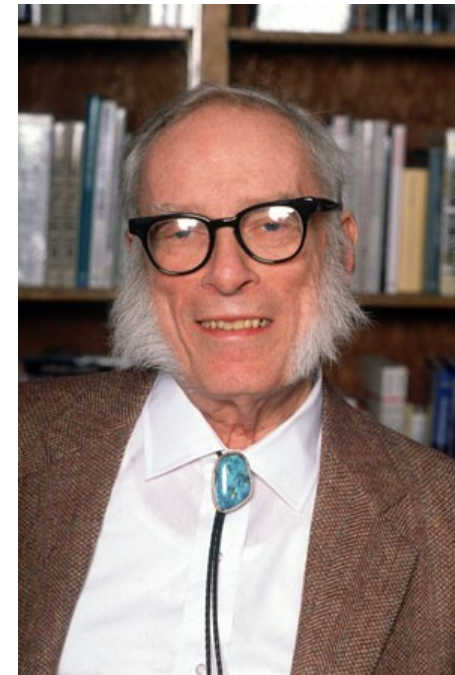
- A sofisticação do software ultrapassou nossa capacidade de construção.
- Nossa capacidade de construir programas não acompanha a demanda por novos programas.
- Nossa capacidade de manter programas é ameaçada por projetos ruins.

# Engenharia de Software

Conceitos,  
elementos e relação  
com Gestão de  
Projetos

***"Parte da desumanidade do computador é que, uma vez que é competentemente programado e trabalhando bem, é completamente honesto."***

***Isaac Asimov***



# Principais Temas em ES

## •Engenharia:

- Análise
- Requisitos
- Design
- Arquitetura
- Interface  
Homem-máquina
- Componentização
- Testes
- Métodos Formais
- Reuso

## •Gestão:

- Processos de Software
- Métricas
- Planejamento
- Análise de Riscos
- Estimativas
- Garantia da Qualidade
- Gestão de Configuração

## •Barry Boehm:

- Engenharia de software envolve a aplicação prática de conhecimento científico para o projeto e construção de programas de computador e a documentação associada necessária para desenvolvê-los, operá-los e mantê-los.

## •IEEE:

- Engenharia de software é a aplicação de uma abordagem sistemática, disciplinada, quantificável para o desenvolvimento, operação e manutenção de software, bem como o estudo destas abordagens, ou seja, a aplicação da engenharia ao software.

# Abordagens



- **Requisitos de software:**

- A elicitação, análise, especificação e validação de requisitos de software.

- **Software design:**

- O projeto de um software é habitualmente feito com a ferramentas CASE, usando padrões de formato, como UML (Unified Modeling Language)

- **Desenvolvimento de software:**

- A construção de software através da utilização de linguagens de programação.

- **Testes de Software**

- **Manutenção de Software:**

- melhorias e correções

- **Qualidade de Software**



# Abordagens

- **Gerenciamento de Configuração de Software:**
  - Devido complexidade dos ambientes, sua configuração (tais como controle de versões, fonte etc.) têm de ser geridos de forma padronizada e estruturada (e automatizada).
- **Gestão de ES:**
  - basicamente gerenciamento de projetos, mas com nuances não encontradas em outras disciplinas.
- **Processo de desenvolvimento de software:**
  - polarização entre os profissionais sobre os principais paradigmas → ágil X cascata
- **Ferramentas de ES:**
  - CASE - Computer Aided Software Engineering
- **Localização:**
  - adaptação do software às diferentes linguagens e culturas.

# Disciplinas Relacionadas



## •Ciência da Computação

- Engenharia de software é considerado um subcampo da Ciência da Computação por muitos acadêmicos. Os fundamentos da engenharia de software vêm dessa ciência.

## •Gestão de Projetos

- A construção de um sistema de software é geralmente considerado um projeto e seu gerenciamento usa muitos dos princípios do campo da gestão de projeto.

## •Engenharia de Sistemas

- Engenheiros de Sistemas lidam com a complexidade de grandes sistemas há muitas décadas e o conhecimento adquirido nessa área é aplicado a muitos problemas da engenharia de software.

# Organizações



- Association for Computing Machinery – ACM

- 1947, 1ª sociedade científica de computação, 83 mil membros em 2007, HQ em New York

- IEEE Computer Society

- 1971, Unidade do IEEE (1963), visão: "ser o principal fornecedor de informações técnicas e serviços para profissionais do mundo da computação", HQ em NY

- Software Engineering Institute

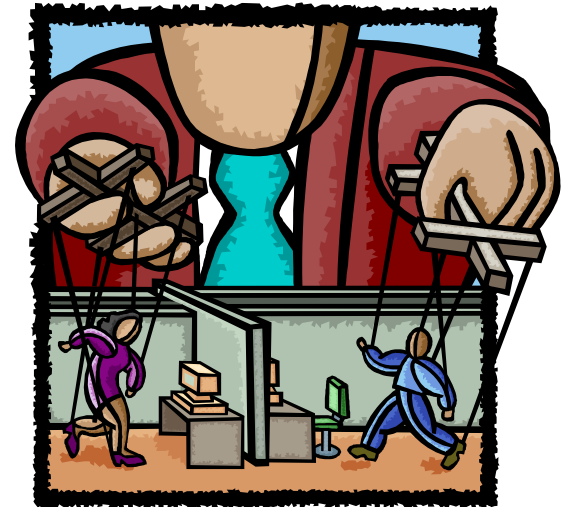
- 1984, Carnegie Mellon University, Pittsburgh, práticas de gestão, engenharia e aquisição, segurança (Security Quality Requirements Engineering - SQUARE)

- British Computer Society

- 1957, mais de 65 mil membros, Questões Governamentais, Jogos, Project Management, Segurança, Gestão de Serviços, SMEs (PME), HQ em Swindon, UK

# Tópicos de Gestão

- Liderança
- Gestão de Recursos Humanos
- Gestão de Projeto
- Gestão de Processos
- Processos



# Tópicos de Gestão



## •Liderança

- Coaching
- Comunicação
- Saber ouvir
- Motivação
- Visão
- Exemplo

# Tópicos de Gestão



## •Gestão de Recursos Humanos

- Contratação
- Retenção
- Staffing
- Treinamento
- Avaliação

# Tópicos de Gestão

## •Gestão de Projeto

- Estabelecimento de metas
- Interação com Cliente e Usuário
- Estimativas
- Gestão de Risco
- Gestão de Mudanças

# Tópicos de Gestão



## •Gestão de Processos

### •Processos

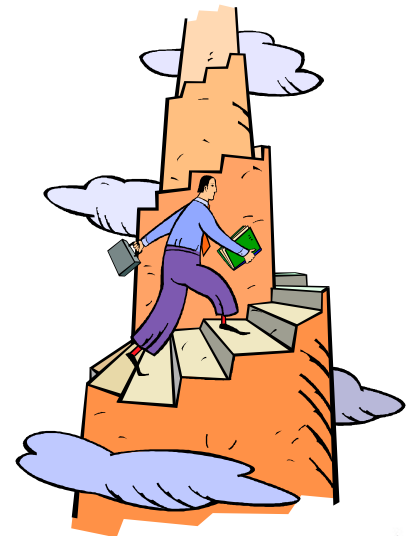
#### •Processos de Desenvolvimento de Software

### •Metricas



# Ciclo de Vida

- Fases do Ciclo de Vida de desenvolvimento
- Fases do Ciclo de Vida do produto
- Releases (estágios)
- Modelos de desenvolvimento



# Ciclo de Vida de Desenvolvimento



- Requisitos (Elucidação / Análise)
- Arquitetura de Software
- Codificação
- Teste (bugs)
  - Black box
  - White box
- Qualidade (conformidade)

# Fases do Ciclo de Vida do Produto



- Concepção
- Protótipo
- Release Inicial
- Releases Menores
- Releases de Correções
- Manutenção
- Obsolescência

# Releases (estágios)

- Alpha
- Beta
- Gold master
- Organização
  - 1.0; 2.0
  - 1.2.3
  - 1.2.3.4

# Modelos de desenvolvimento

- Modelo Cascata
- SSADM (Structured Systems Analysis and Design Method)
  - Modelo Lógico de dados
  - Modelo de fluxo de dados
  - Modelo de comportamento de entidades
- Modelo Espiral (iterativo)
- V-model (Test Driven)
- Desenvolvimento Ágil
- DSDM (Dynamic Systems Development Method)
  - Interativo, ágil, RAD, pequenos projetos
- Chaos model — Chaos strategy
  - Extensão do cascata e espiral
  - "TUDO" deve ser definido, implementado e integrado



# Paradigmas de Programação

- Os paradigmas da programação dependem da plataforma (linguagem)

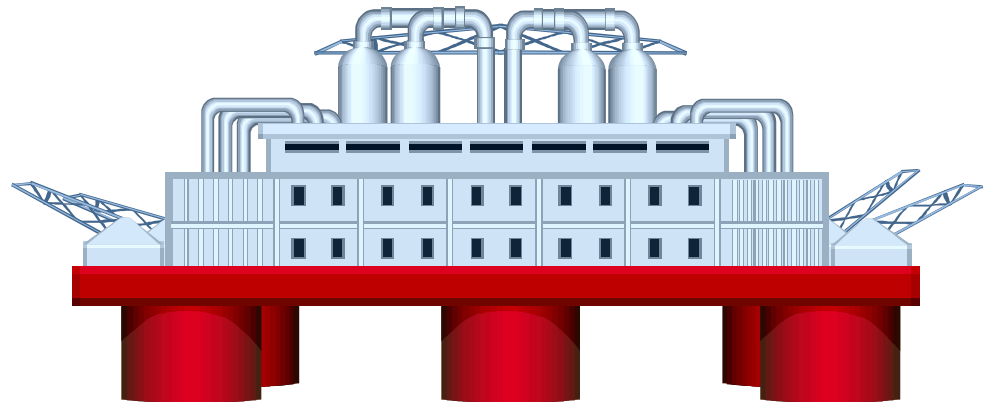
- Decomposição Funcional
- Programação estruturada
- Orientação a Objetos
- Orientada a Aspectos
  - Post-object programming



# Plataformas

- Plataformas combinam HW, SW e OS.  
Cada vez mais poderosas e baratas  
→ democratização das ferramentas e aplicações.

- BREW (Binary Runtime Environment for Wireless)
- Supercomputadores (CRAY, IBM, HP)
- Mainframes
- Minicomputadores
- Windows PCs
- Mac OS
- PCs Linux
- Palm PDAs
- Proprietários
  - Stratus
  - Symbian OS



# Para Que?

- Entregáveis  
(Os Engenheiros de software NÃO são responsáveis por todos os entregáveis)

- Aplicação (O software)
- Banco de dados (Estruturas)
- Documentação
  - Políticas
- Migração
- Treinamento
- Suporte
  - White papers – explica as tecnologias usadas nas aplicações
- Marketing
  - Benckmarking





# Perguntas que Engenharia de Software quer responder:

- Porque demora tanto para concluir um projeto (não cumprimos prazos)?
- Porque custa tanto (uma ordem de magnitude a mais)?
- Porque não descobrimos os erros antes de entregar o software ao cliente?
- Porque temos dificuldade de medir o progresso enquanto o software está sendo desenvolvido?

# Causas óbvias

- Não dedicamos tempo para coletar dados sobre o desenvolvimento do software - resulta em estimativas “a olho”.
- Comunicação entre o cliente e o desenvolvedor é muito fraca.
- Falta de testes sistemáticos e completos.

# Causas menos óbvias

- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico (característica 1).
- Gerentes sem *background* em desenvolvimento de SW.
- Profissionais recebem pouco treinamento formal.
- Falta investimento (em ES).
- Falta métodos e automação.

# Mitos do Software - Administrativos



- Um manual oferece tudo que se precisa saber.
- Computadores de última geração solucionam problemas de desenvolvimento.
- Se estamos atrasados, basta adicionar programadores e tirar o atraso

# Mitos do Software - do Cliente

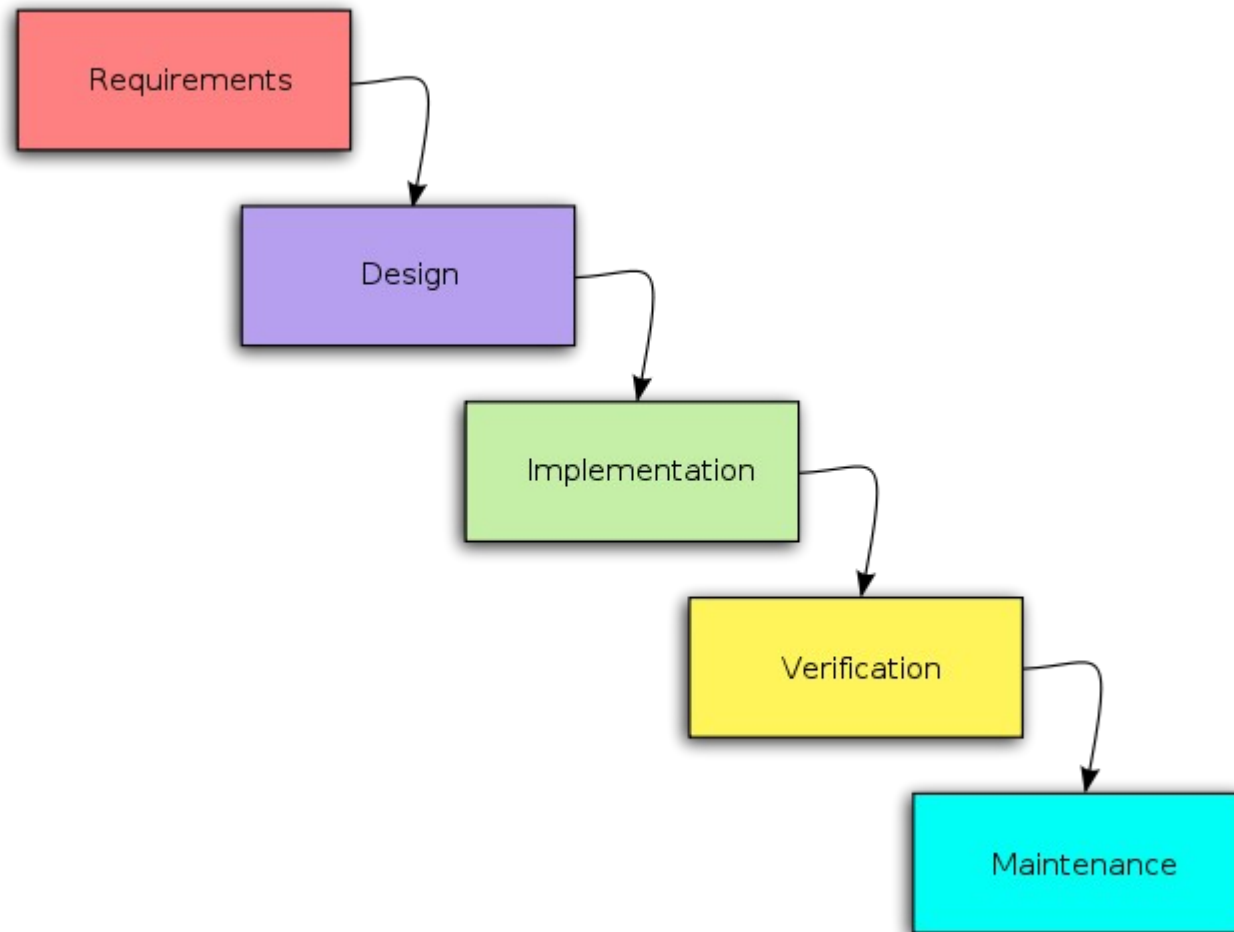


- Uma declaração geral é suficiente para começar a escrever programas.
- Mudanças podem ser facilmente acomodadas em um projeto.

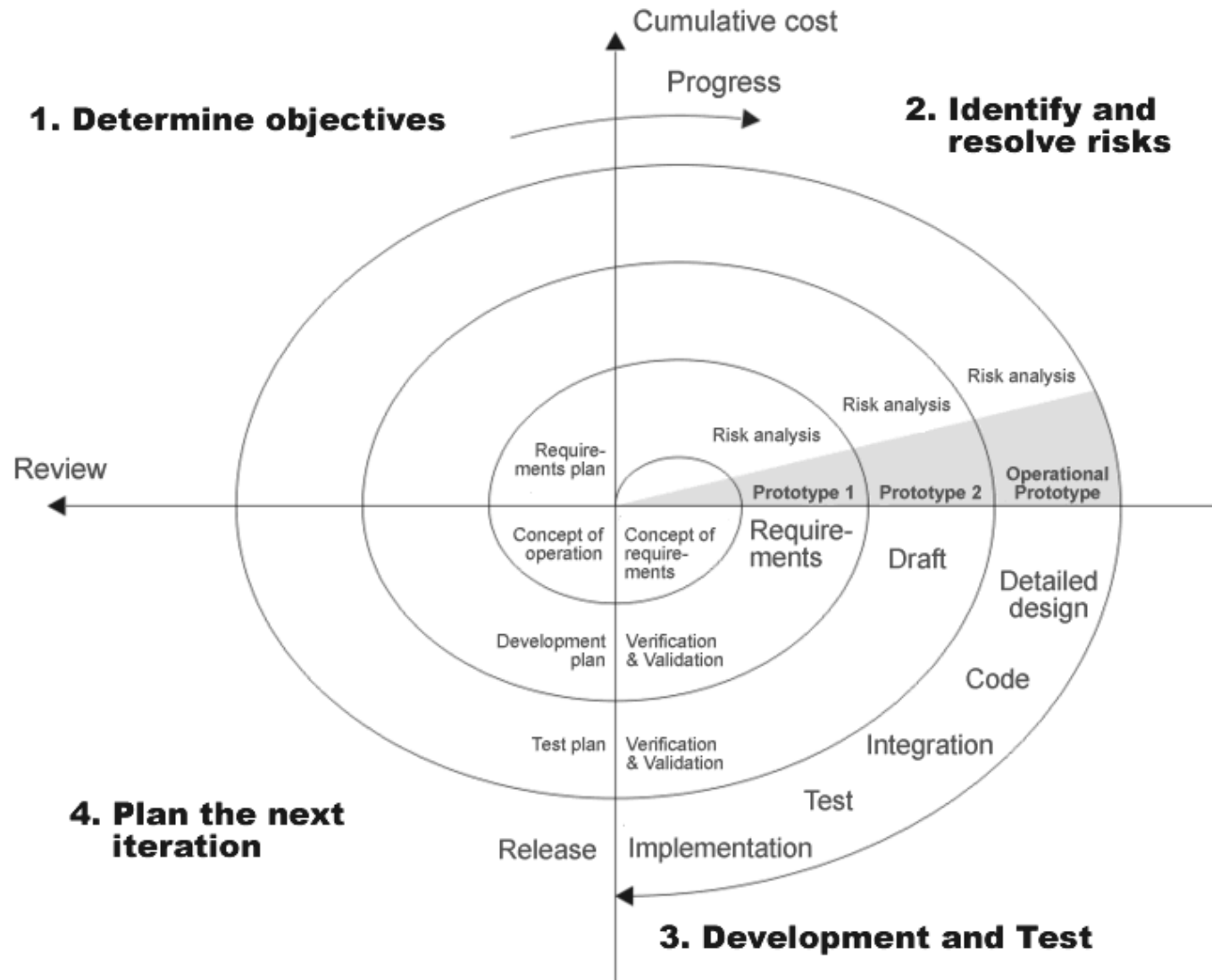
# Mitos do Software - do Profissional

- Um programa está terminado ao funcionar.
- Quanto mais cedo escrever o código, mais rápido terminarei o programa.
- Só posso avaliar a qualidade de um programa em funcionamento.
- A única coisa a ser entregue em um projeto é o programa funcionando.

# Waterfall Model

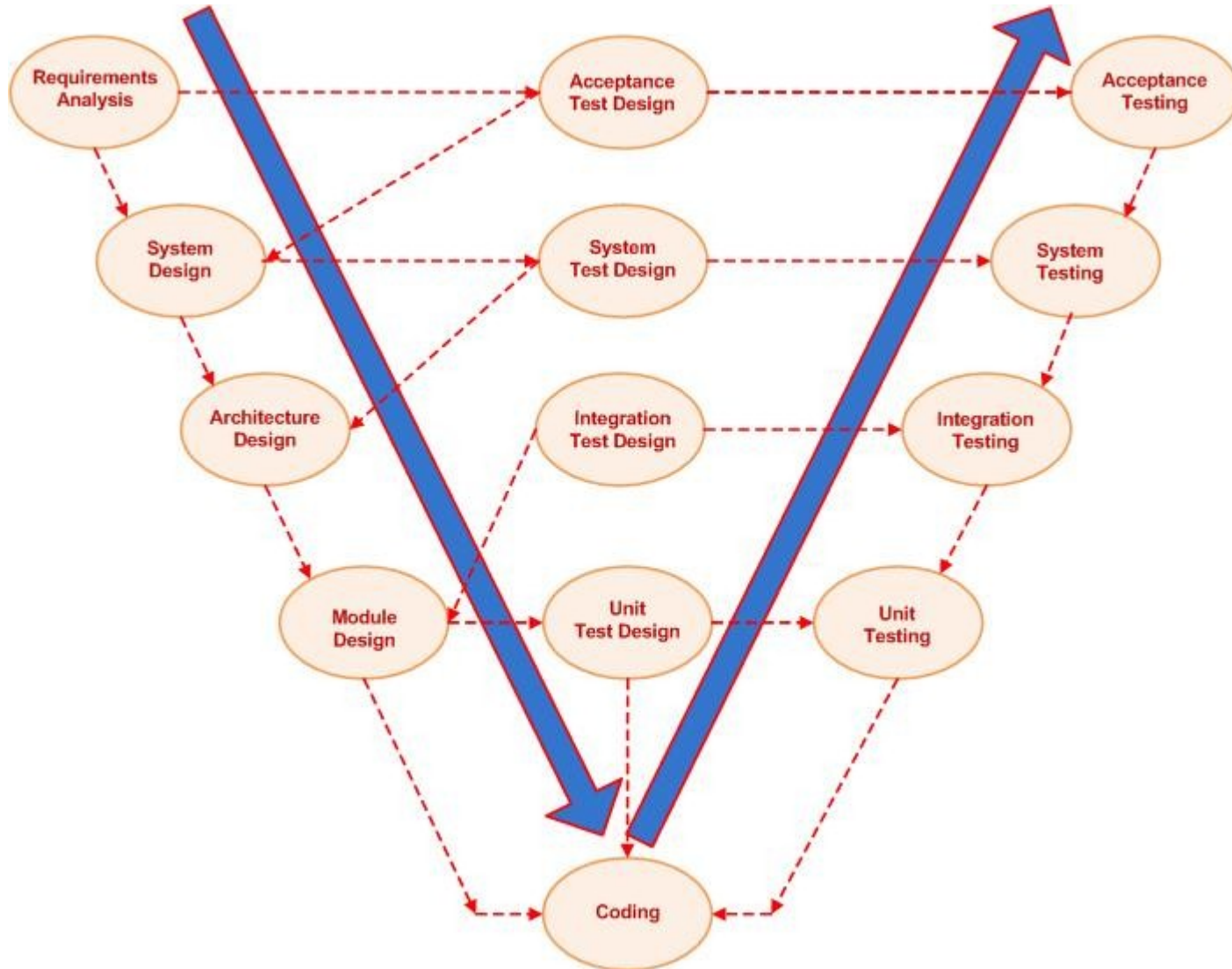


# Spiral model - Boehm 1988

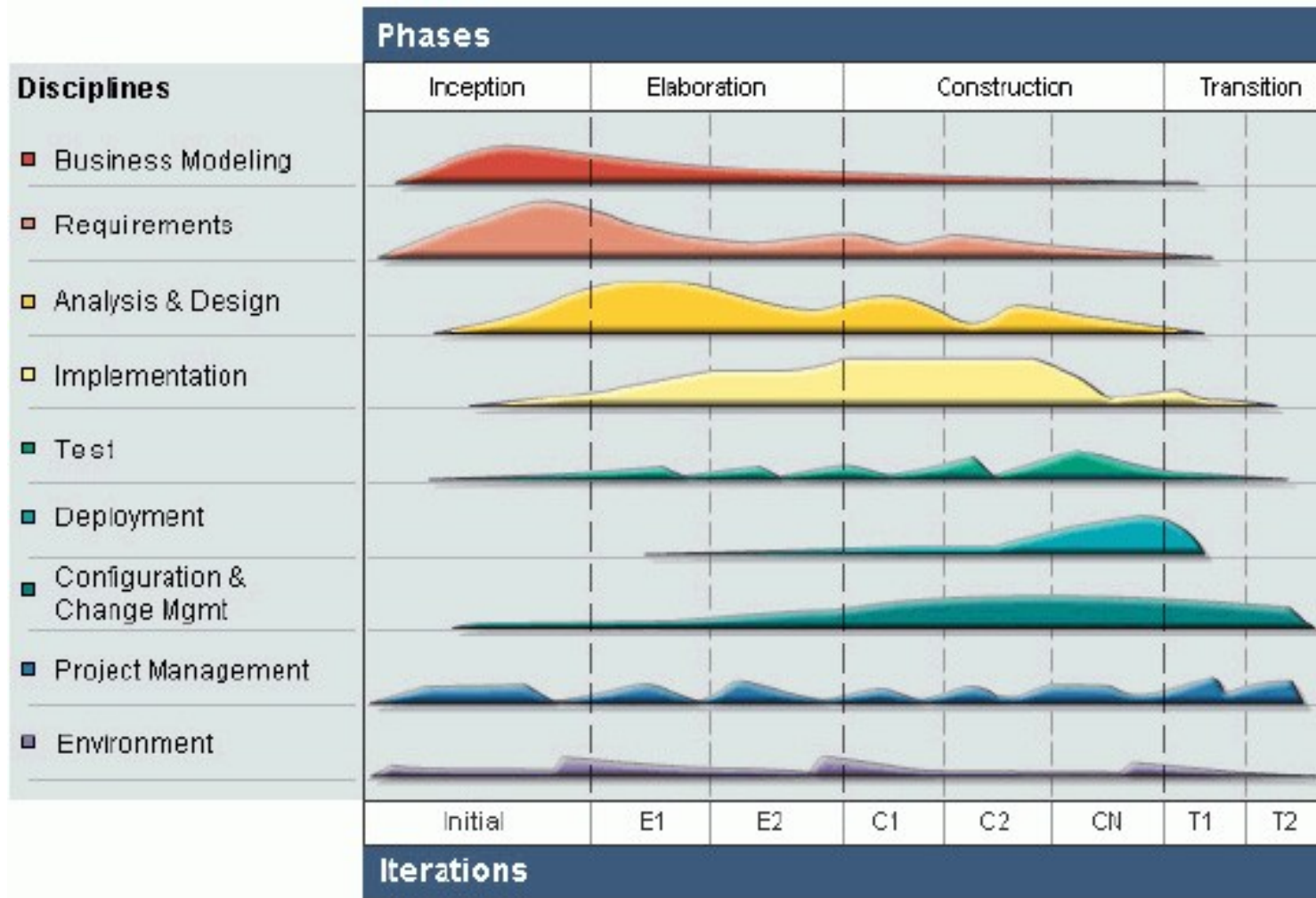




# V-model



# IBM Rational Unified Process (RUP)



# Desenvolvimento Ágil



## •Métodos

- Agile Modeling
- Agile Unified Process (AUP)
- Agile Data Method
- DSDM
- Essential Unified Process (EssUP)
- Extreme programming (XP)
- Feature Driven Development (FDD)
- Getting Real
- Open Unified Process (OpenUP)
- Scrum

## •Práticas

- Test Driven Development (TDD)
- Behavior Driven Development (BDD)
- Continuous Integration
- Pair Programming
- Planning poker

# Desenvolvimento Ágil

- Inicialmente, métodos ágeis eram conhecidos como métodos leves.
- Em 2001, membros proeminentes da comunidade se reuniram em Snowbird e adotaram o nome métodos ágeis = Manifesto ágil - [www.manifestoagil.com.br](http://www.manifestoagil.com.br)
- 12 princípios e práticas desta metodologia
- Os métodos ágeis iniciais:
  - Scrum (1986)
  - XP (1996)
  - Crystal Clear
  - Feature Driven Development
  - Entre outras...

# Manifesto Ágil

**Indivíduos e  
interação entre eles**

mais que

Processos e ferramentas

**Software em  
funcionamento**

mais que

Documentação abrangente

**Colaboração com  
o cliente**

mais que

Negociação de contratos

**Responder a  
mudanças**

mais que

Seguir um plano

# Encerrando nossa aula



- Nesta aula, tratamos sobre:

- Software

- Estudamos a história do software, a criação e evolução de seus conceitos. Como chegamos ao estágio que estamos hoje.
- Entendendo por quê um software não pode ser comparado a produtos tangíveis, requerendo um tratamento diferenciado quando da gestão de projetos para sua fabricação.
- Compreendendo os impactos da Crise do Software em nossos projetos atuais.

- Engenharia de Software

- Conhecemos melhor a disciplina, sua aplicabilidade e sua motivação.
- Conteúdo da matéria: abordagens, relacionamento com outras disciplinas.
- Entendendo como a Engenharia de Software pode ajudar gestores de projetos de software

# Próxima Aula

## • Modelos de Processo de Software

- Cascata
- Espiral
- RUP
- Ágil (XP e Scrum)

## • Workshop sobre Ciclos de Vida de Projeto

## • Fábricas de Software

- Conceitos

