História da Engenharia de Software

Evolução Histórica da Computação e Aplicações

Profa. Rosana Braga

rtvb@icmc.usp.br

Introdução

"O mundo de hoje não poderia viver sem o software"

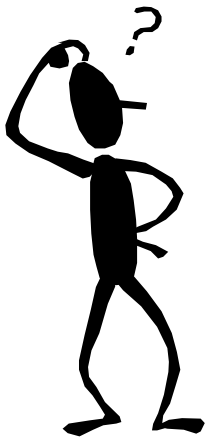
Sommerville, 2011

Introdução

- Tudo o que nos rodeia envolve software: desde produtos eletrônicos, serviços de transporte, médicos, telecomunicações, militar, industrial e financeiro, entretenimento, educação, etc.

Introdução

O que é **software?**



O que é software?

- Mais do que programas-fonte e executáveis: compreende também a documentação associada a ele, dados e configuração
- Tudo isso impacta a futura evolução e manutenção do software

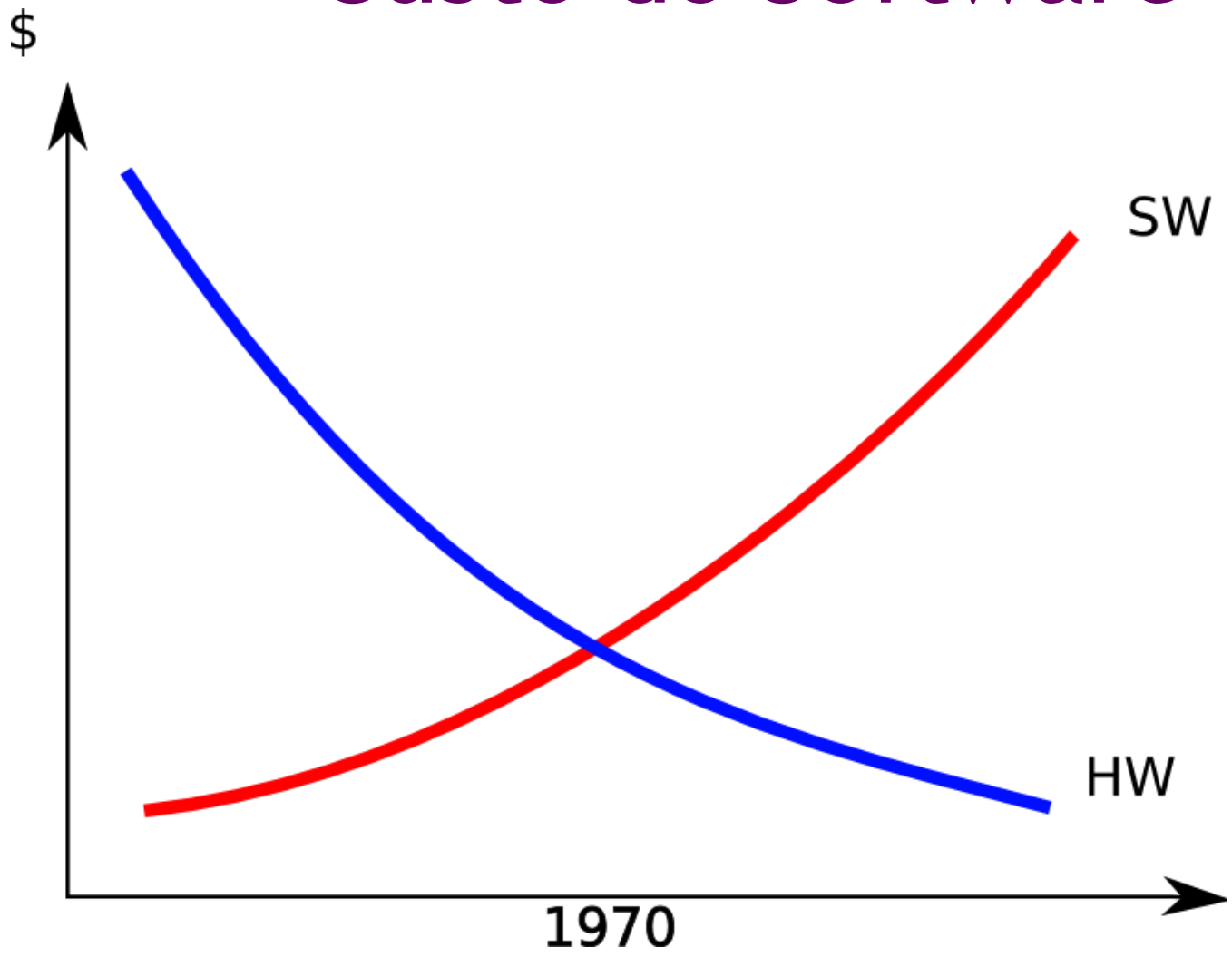
Introdução

- Lei das “**consequências não pretendidas**” (Pressman, 2006)
  - Ex: isopor, penicilina, microondas, facebook...
  - Na década de 50 não era esperado que o software tomasse as proporções atuais
  - Quais as consequências disso?

Introdução

- Problemas com o software:
  - Software que não funciona
  - Altos custos
  - Prazos não cumpridos
  - Baixa qualidade:
    - cheios de defeitos
    - confiabilidade duvidosa
    - difíceis de usar
    - lentos,
    - não portáveis,
    - etc...

Custo do Software



92

9/9

0800 Arctan started { 1.2700 9.037 847 025
1000 " stopped - arctan ✓ 9.037 846 995 correct
13 sec (032) MP - MC ~~1.9821447000~~ ~~2.13047676415 (033)~~ 4.615925059(-2)
(033) PRO 2 2.13047676415
correct 2.13067676415
Relays 6-2 in 033 failed special speed test
In relay " 11.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.

Relay 3145
Relay 3370

1545 

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

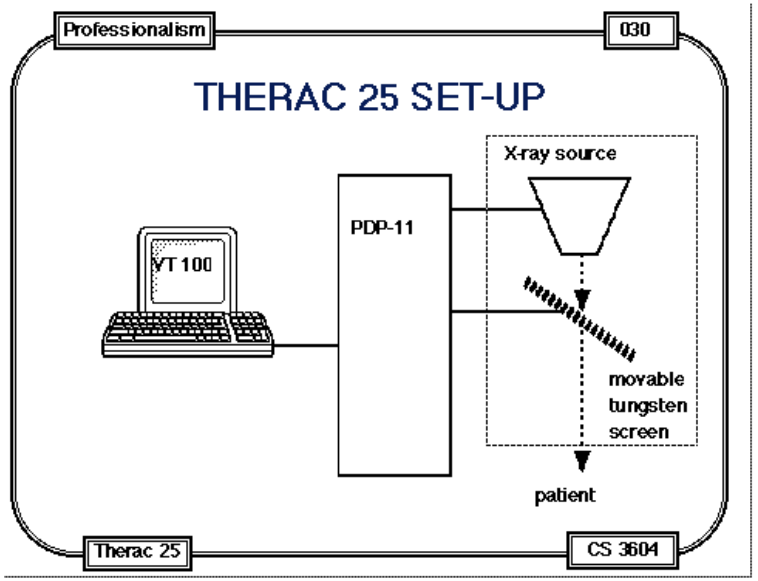
~~1645~~ 1630 Arctangent started.
1700 closed down.

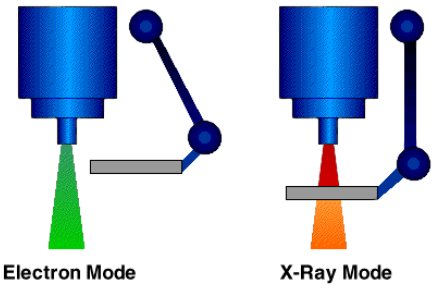
Bugs históricos

- **Mariner I – 1962**
  - Missão observar planeta Vênus
  - Fórmula matemática foi equivocadamente transcrita para o computador
  - Desviou do curso e foi destruído 4 min após lançamento
  - Prejuízo: US$ 18,5 mi

Bugs históricos

- **Therac-25 – 1985/1987**
  - Dispositivo de terapia por radiação sobre células cancerosas
  - Libera doses letais de radiação em vários consultórios médicos
  - Condição de disputa no SO
  - 5 mortes, várias pessoas feridas





Bugs históricos

- **Divisão de pontos flutuantes nos processadores Pentium da Intel – 1993**
  - Erro em divisões dentro de uma faixa de números (erro ~0,006% no arrendondamento)
  - 3 a 5 milhões de peças com defeito
  - *Recall* para todos que quiseram trocar
  - Custou à Intel US$ 475 milhões

$$\frac{4195835}{3145727} = 1.333820449136241002$$

$$\frac{4195835}{3145727} = 1.333739068902037589$$

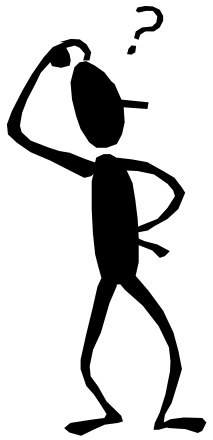
Bugs históricos

- **Ariane 5 vôo 501 – 1996**
  - Levou uma década de desenvolvimento e custou 7 bilhões de dólares.
  - Foguete com código reutilizado do Ariane 4 (outro hardware);
  - *Overflow* de inteiro: conversão de *float* de 64-bits para inteiro 16-bits com sinal;
  - O processador primário do foguete sobrecarregou os motores que se desintegraram em 40 segundos;
  - Não tripulado (sem vítimas); prejuízo de US$ 370 milhões

Bugs históricos

- **Bug do milênio (Y2K) – 2000**
  - Datas com apenas 2 dígitos para o ano
  - Uma das maiores histerias da história
  - Ao virar o ano 2000, a preocupação era que contasse como 1900
  - Entre US$ 300 e US$ 500 bi no mundo todo

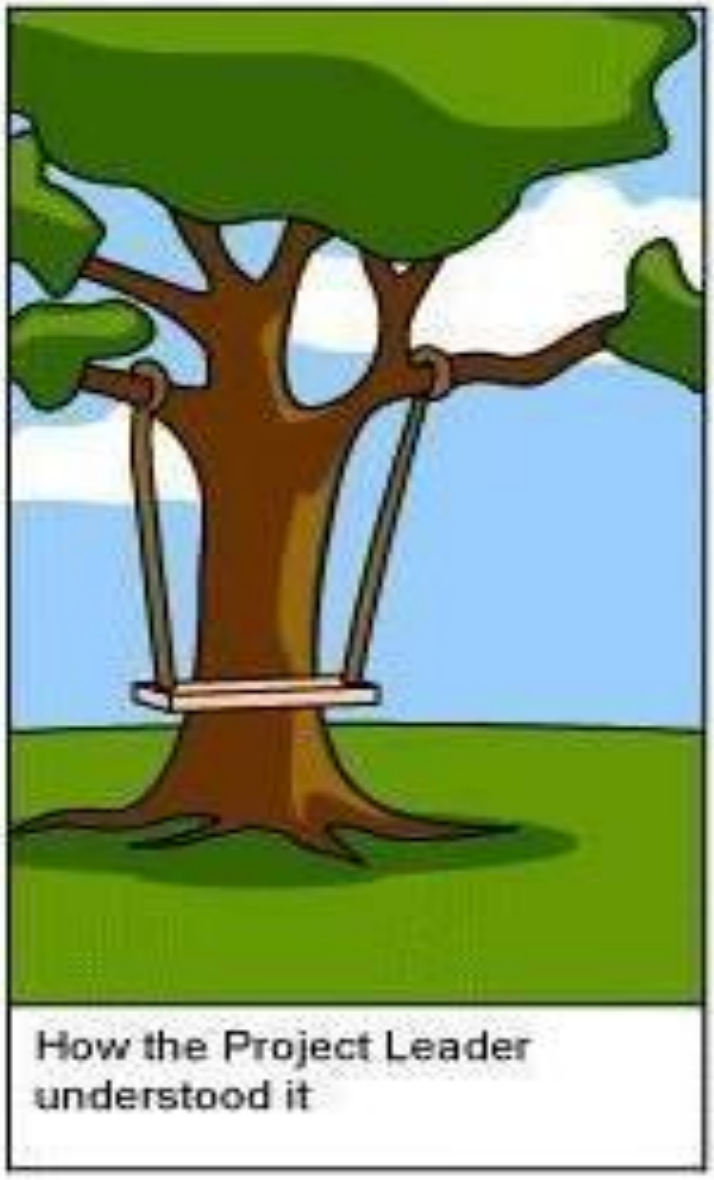
Porque tantos bugs ocorrem?



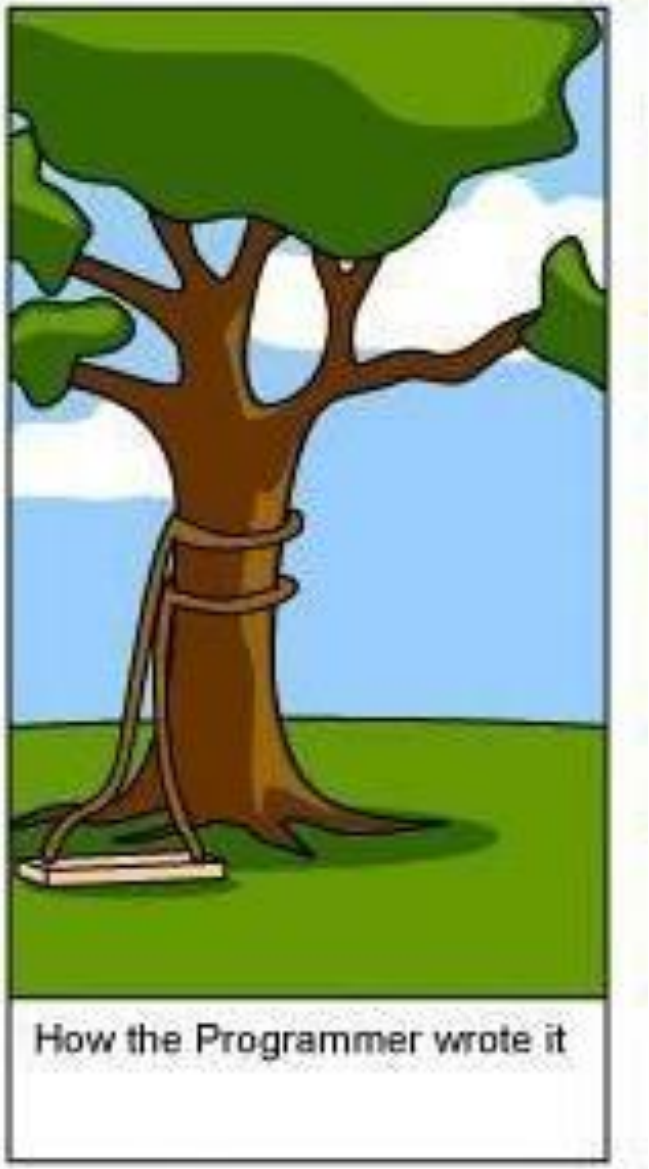
Porque tantos bugs ocorrem?

- Falta de testes?
- Projetos mal feitos?
- Falta de controle de qualidade?
- O cliente não sabe o que quer?
- Os desenvolvedores não entendem o que o usuário quer?
- Etc. etc etc

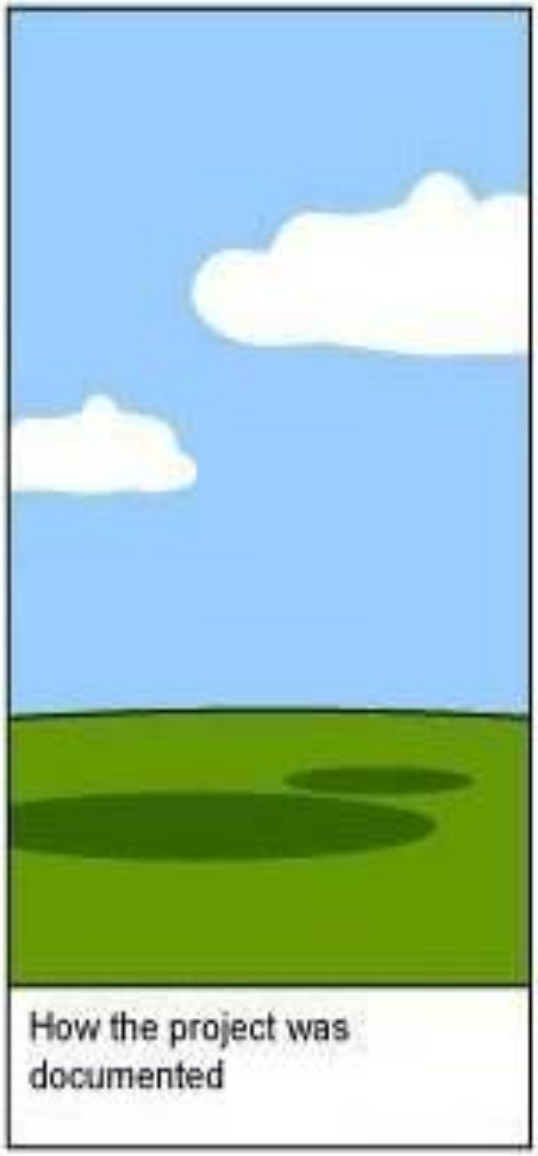

How the customer explained it

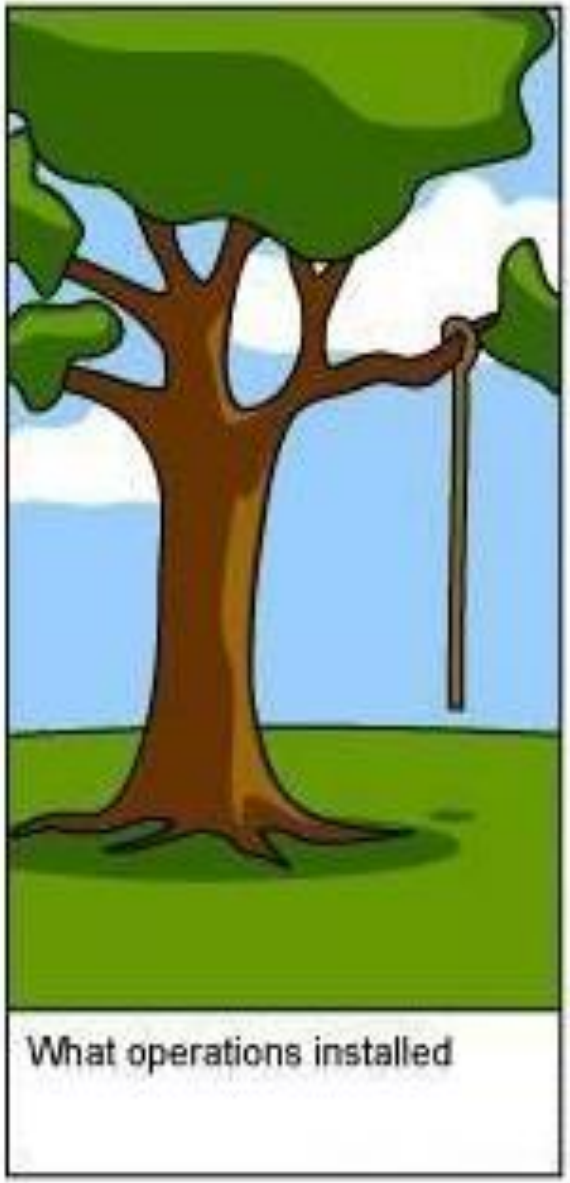

How the Project Leader
understood it

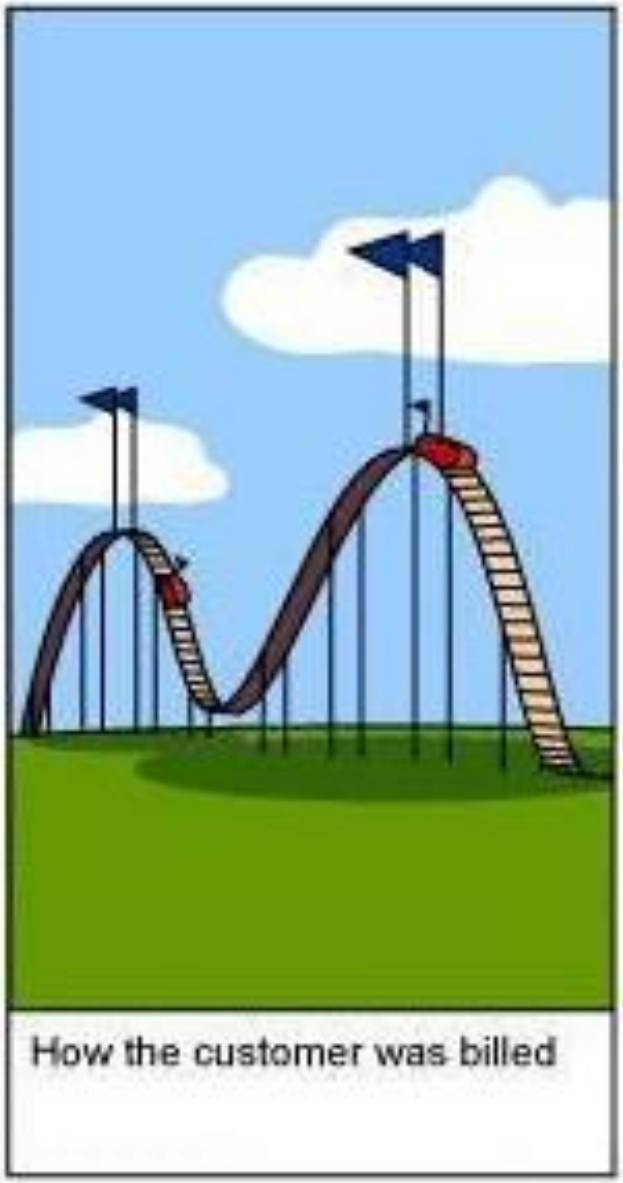

How the Analyst designed it

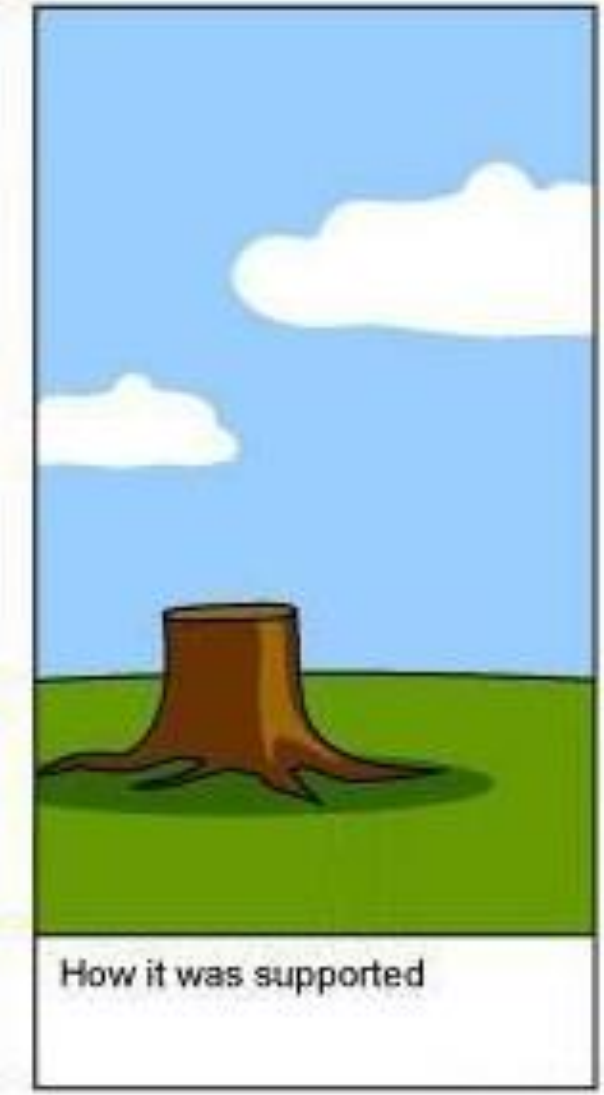

How the Programmer wrote it

How the Business Consultant described it

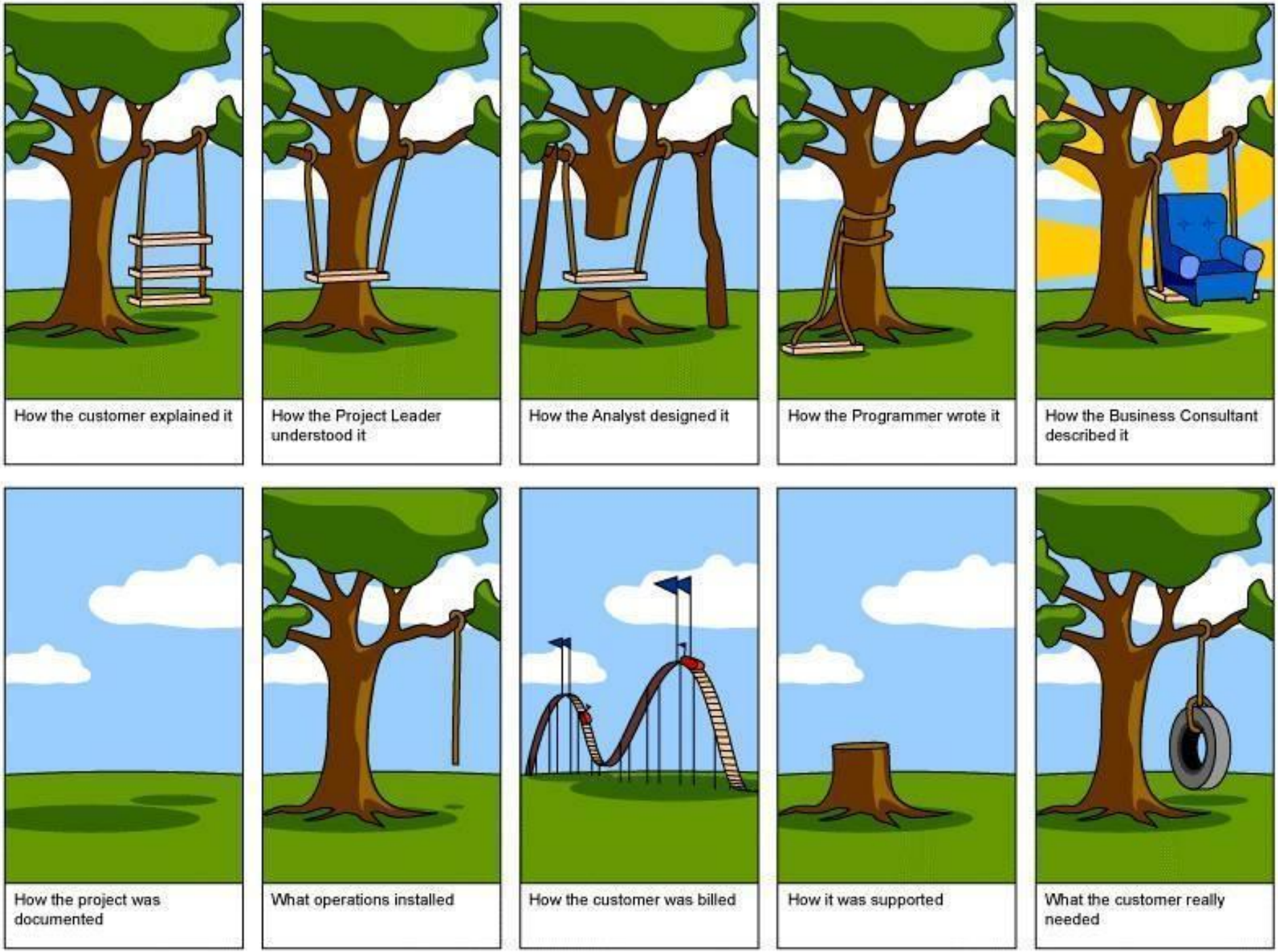

How the project was documented


What operations installed


How the customer was billed


How it was supported

What the customer really needed


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Alguns marcos importantes da Engenharia de Software

- 1950 a 1968: primeiros tempos: desenvolvimento ad hoc
- 1968: Crise do software
  - Termo Engenharia de Software oficialmente definido
- 1970: Modelo Cascata (Royce)
- 1974: Análise/projeto estruturado (Stevens, Myers, Constantine, de Marco e Yourdon)
- 1975: O mítico homem-mês (Fred Brooks)
- 1980 a 2000: desenvolvimento das várias áreas de ES (por exemplo, planejamento de projeto, estimativas de software (COCOMO), vários modelos de processo e modelos de qualidade, teste de software, ferramentas e ambientes de desenvolvimento, etc)

Alguns marcos importantes da Engenharia de Software

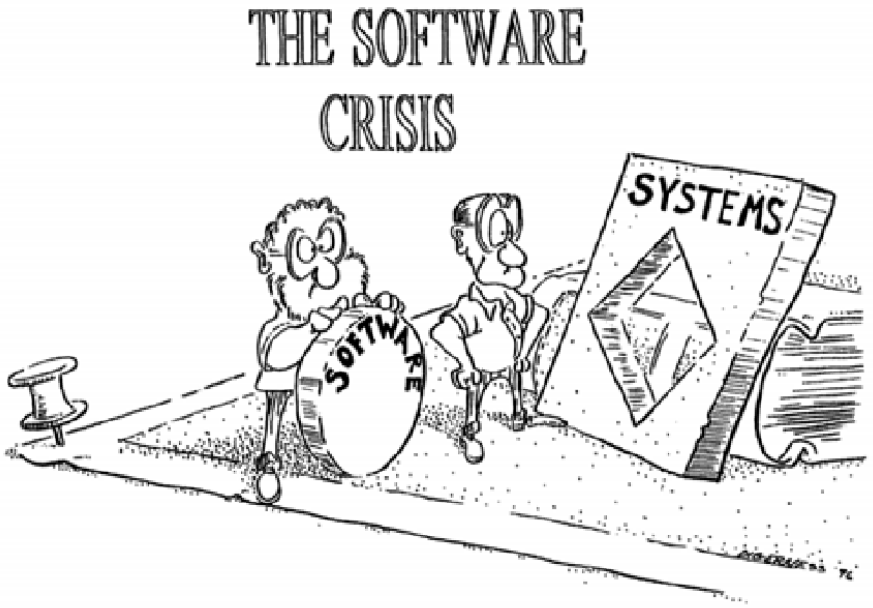
- Fim dos anos 80: surgimento da análise orientada a objetos
- 1997: UML (Unified Modelling Language)
- 2001: O manifesto ágil
- 2002 em diante: vários processos ágeis, foco na interação com o cliente e facilidade de evolução

1950 a 1968: primeiros tempos: desenvolvimento ad hoc

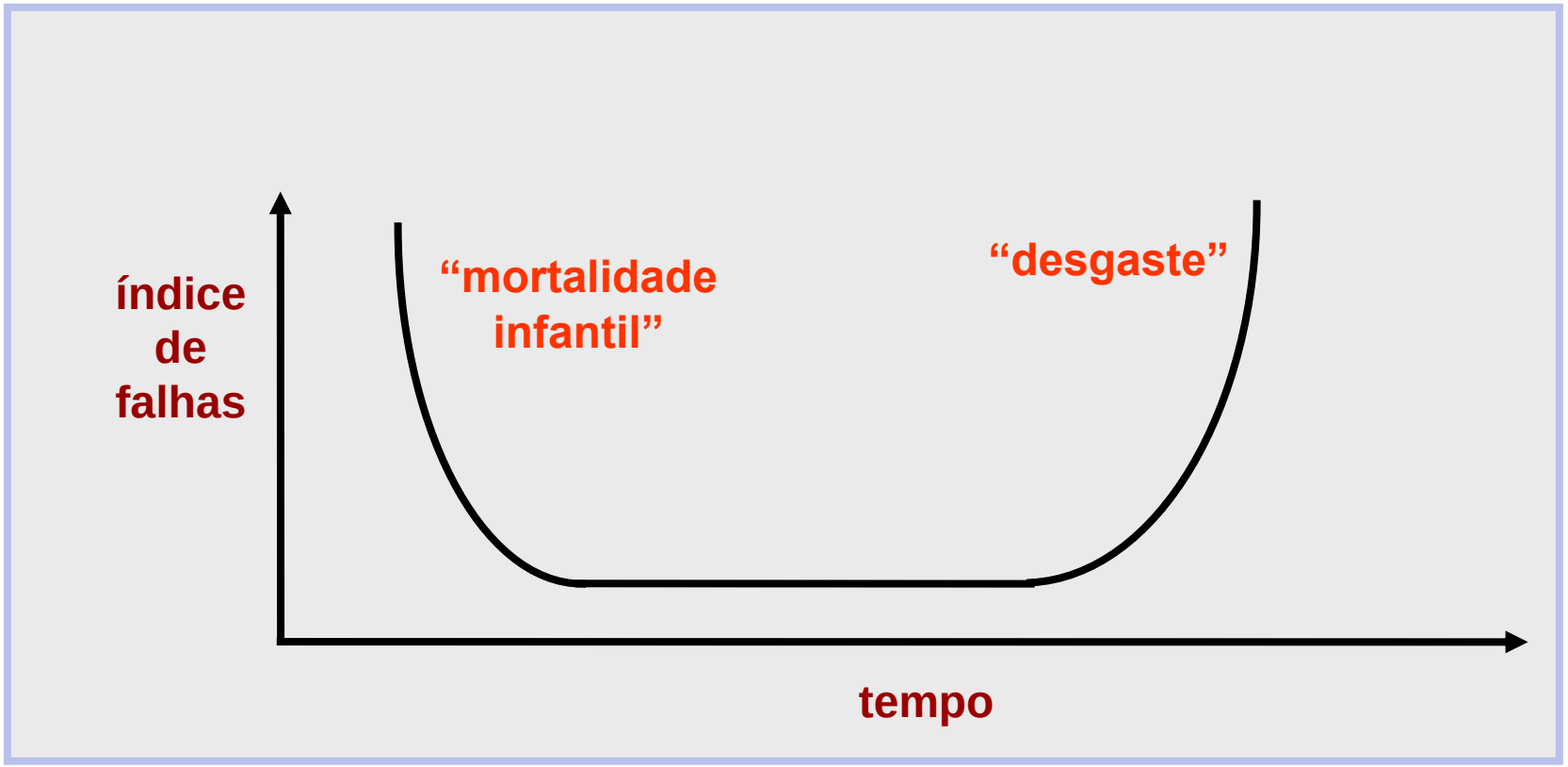
- Software era um conceito abstrato
- Não havia nenhum direcionamento para sua produção, visto que não se encaixava em nenhuma das engenharias conhecidas
- Software feito de maneira “ad-hoc”, isto é, sem nenhum processo associado
- Codificação, na maioria das vezes, era feita diretamente sem nenhuma análise prévia
  - Fluxogramas e documentação textual

Evolução do Software

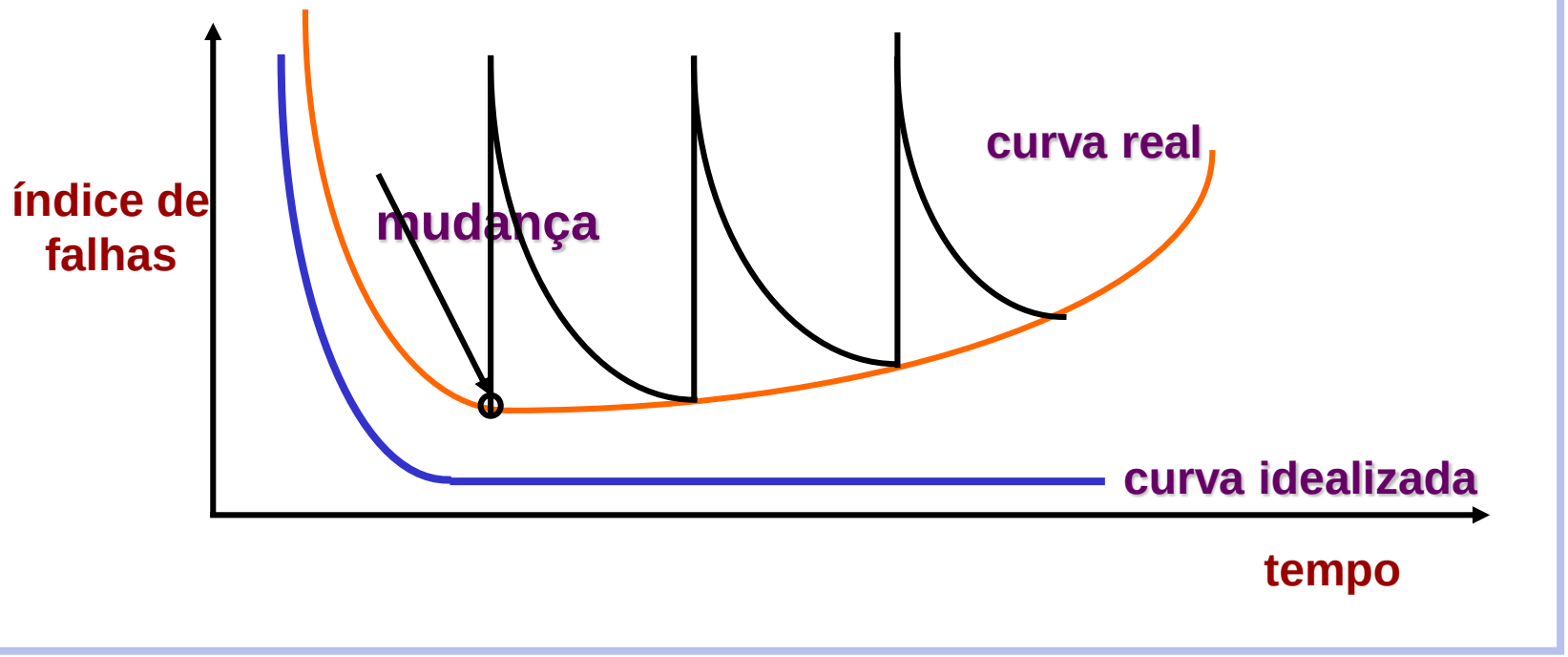
1968 - Crise do Software



Curva de falhas para o Hardware



Curva de falhas do Software



Causas da crise de software

- Projetos que estouram o cronograma
- Projetos que estouram o orçamento
- Baixa qualidade
- Clientes insatisfeitos
- Produtos difíceis de manter....
- e
- Mitos...

Crise do Software

- 1968: 1ª conferência de engenharia de software da OTAN (Organização do Tratado do Atlântico Norte)

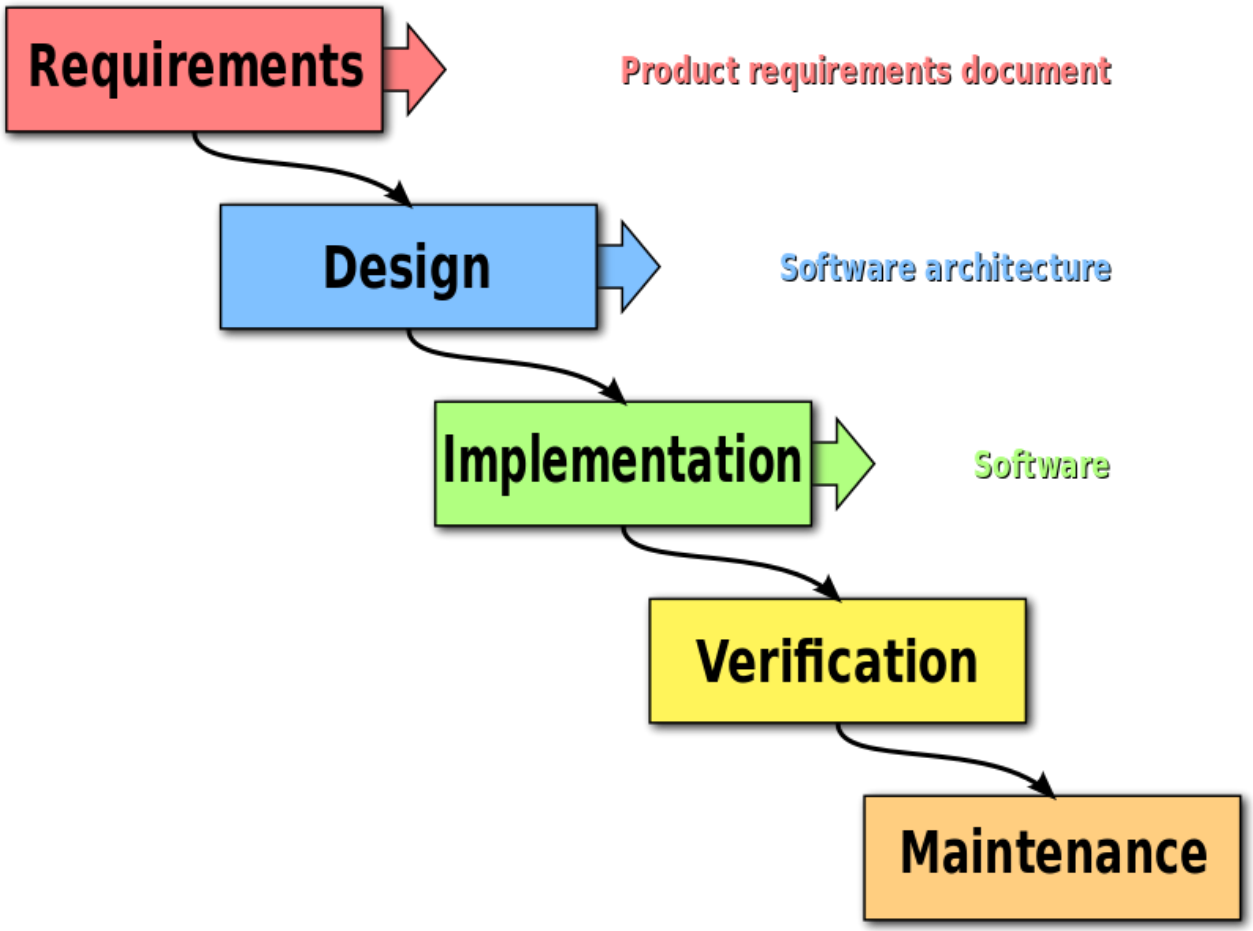
Crise do Software

- Termo crise de software usado por muito tempo para se referir aos problemas do software não acompanhar o desenvolvimento do hardware
- Reconhecimento da falta de técnicas e processos adequados para o desenvolvimento de software de qualidade
- Afirmação de Djikstra (+-1970): especificações de software complexas e obscuras, que tornam o programador “as mais infelizes criaturas da face da Terra”.

1970: O Modelo Cascata (**Winston Walker Royce**)

- modelo mais antigo e o mais amplamente usado da engenharia de software
- modelado em função do ciclo da engenharia convencional
- requer uma abordagem sistemática, seqüencial ao desenvolvimento de software
- o resultado de uma fase se constitui na entrada da outra

O Modelo Cascata



Modelo original, surgiram várias outras versões ao longo dos anos

Análise estruturada (1974 a 1979)

- 1974: Artigo de Stevens, Myers, Constantine sobre projeto estruturado
- Alguns anos depois: Tom de Marco e Yourdon: metodologia para construir software de maneira sistemática, partindo dos requisitos, passando pela modelagem, projeto e programação.
  - Divulgada pelo livro de Gane e Sarson
- A Análise Estruturada utiliza as seguintes ferramentas:
  - DC – Diagrama de Contexto DFD – Diagrama de Fluxo de Dados Dicionário de Dados Descrição de Cenários, Português Estruturado e outros Tabelas de Decisão Árvores de Decisão

1975: O Mítico homem-mês (Brooks)

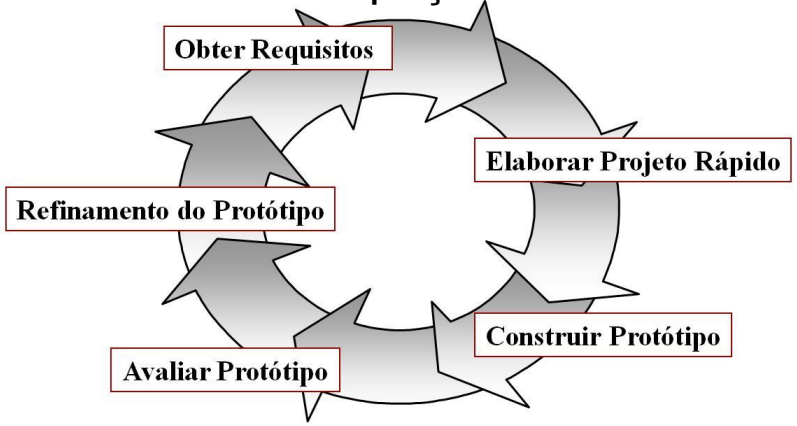
- Em meados dos anos 60, Brooks era gerente do desenvolvimento do sistema operacional do computador mainframe da IBM /360.
- Um dos sistemas mais complexos desenvolvido na época
- Com base nessa experiência, escreveu o livro
- 20 anos depois republicou o livro discutindo as mudanças 20 anos depois

1975: O Mítico homem-mês (Brooks)

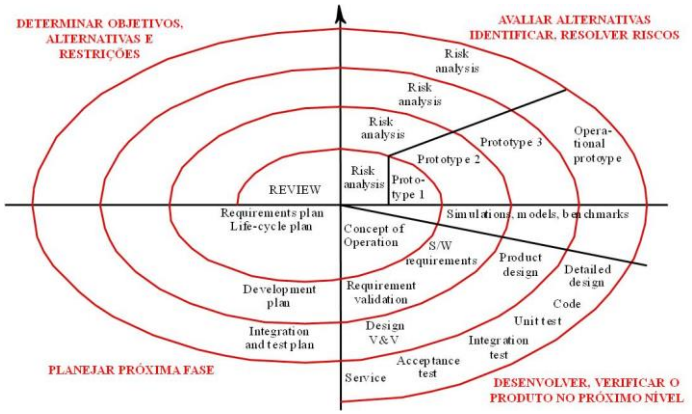
- Lei de Brooks: Acrescentar mais desenvolvedores em um projeto de software atrasado só faz com que ele atrase mais ainda!!

1980 a 2000: criação de vários processos de desenvolvimento

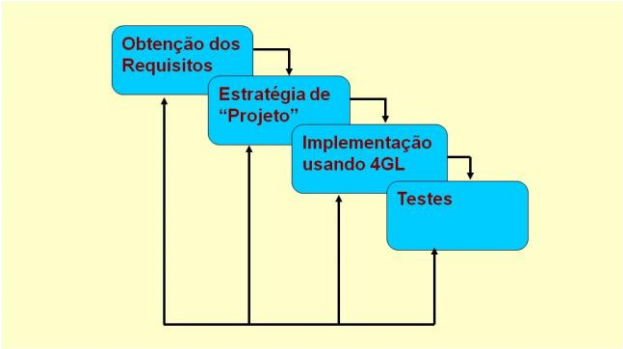
Prototipação



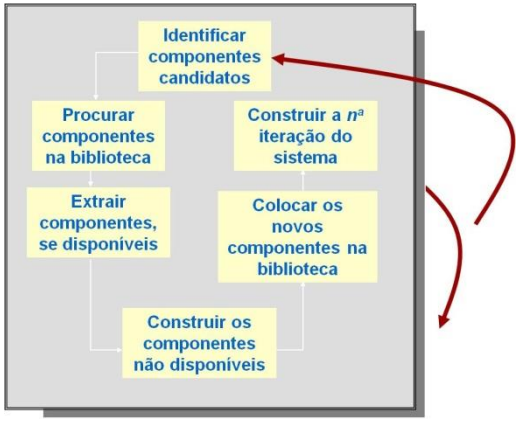
Espiral



Técnicas de 4ª geração

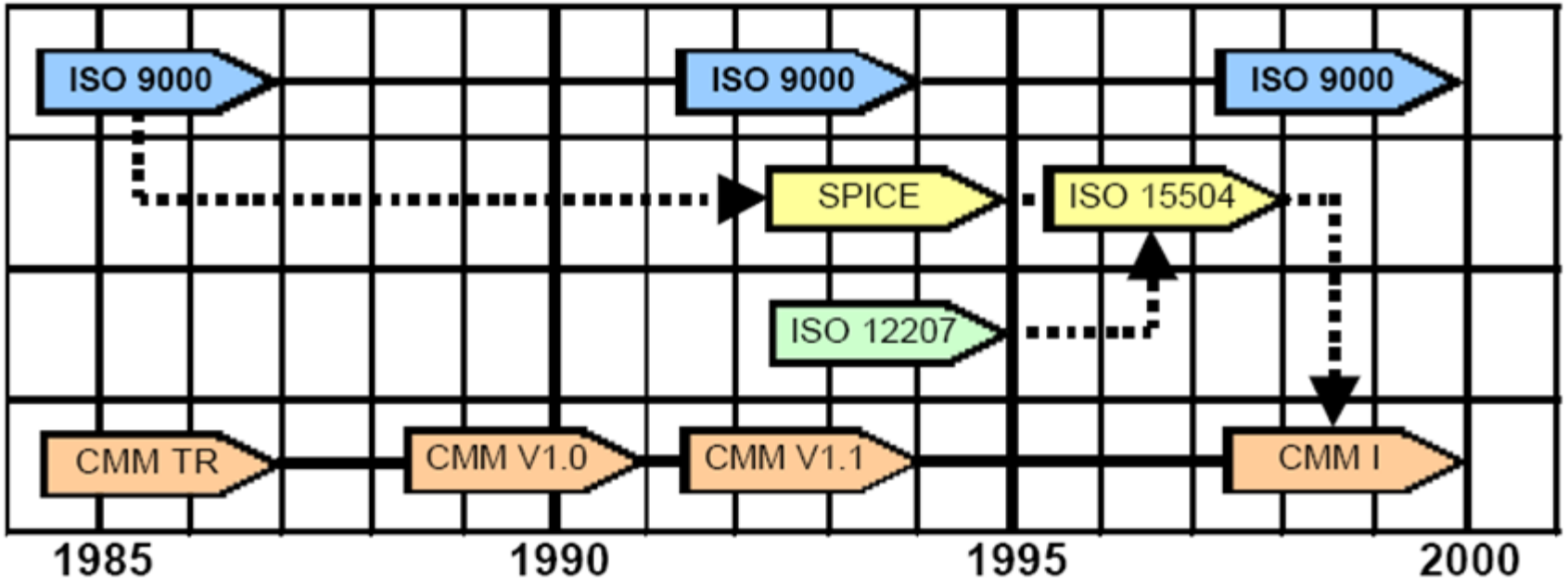


Desev. Baseado em componentes



E outros...

1980 a 2000: modelos de qualidade



Fonte: Sheard, A. Sarah. The Framework Quagmire, A Brief Look. SPC – August 1997

E outros...

1980 a 2000: evolução das diversas áreas da engenharia de software

- Engenharia de Requisitos
- Planejamento de projeto
  - PMBOK
  - Estimativas de software (COCOMO)
- Teste de software
- Ferramentas e ambientes de desenvolvimento
- Geração automática de código e testes
- Gerenciamento de Configuração de Software
- etc

1980 a 2000: evolução das diversas áreas da engenharia de software

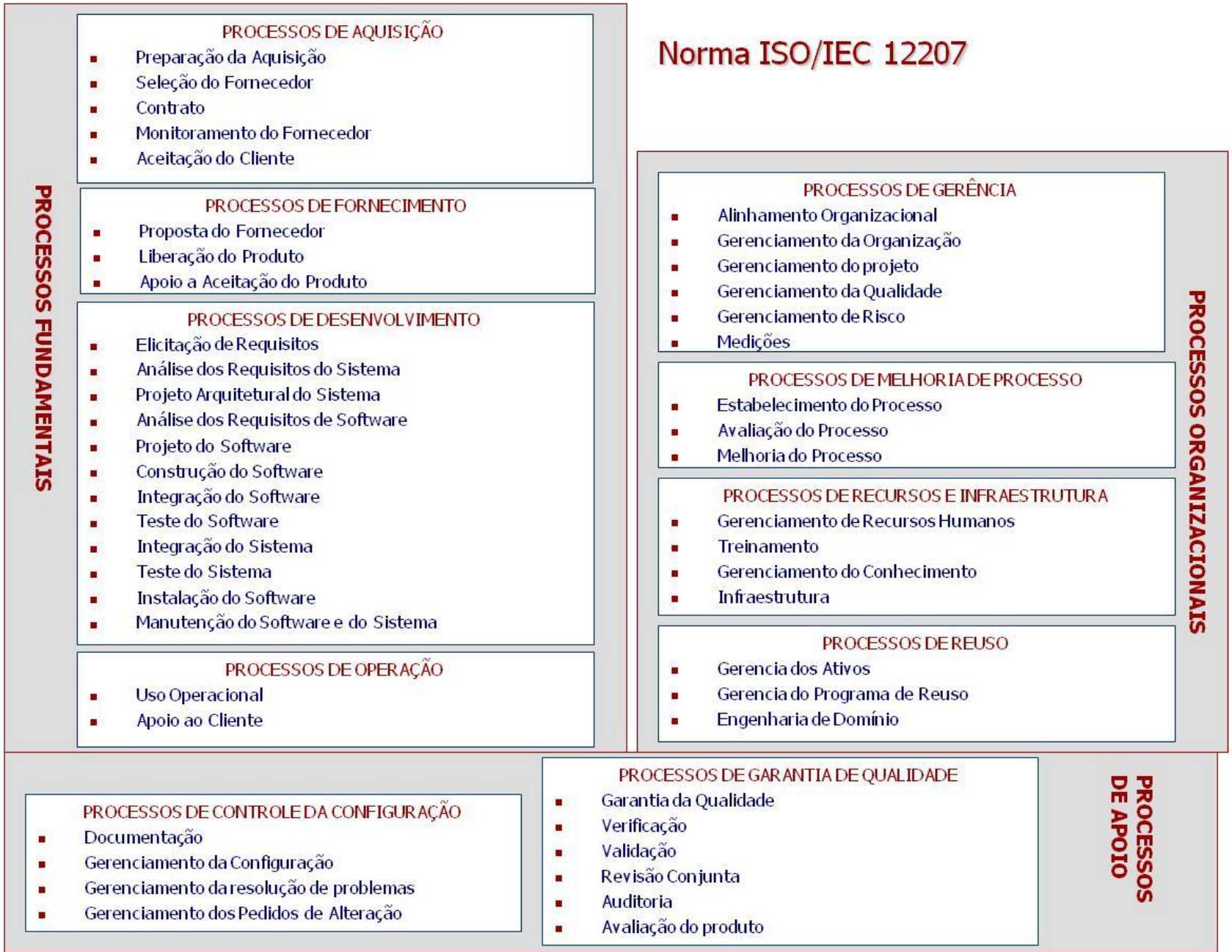
- No ICMC: Laboratório de Engenharia de Software
- Sub-áreas de pesquisa:
  - Methods, techniques, processes and tools for software development:
    - Software architecture, reference architecture and software engineering environment.
    - System of Systems.
    - Software reuse
      - Software patterns, components, frameworks
      - Model Driven Development
      - Software product lines, application generators etc.
      - Service-oriented applications.
      - Object and aspect-oriented programming.

(continua)

1980 a 2000: evolução das diversas áreas da engenharia de software

- Software testing
  - Functional, structural and mutation testing
  - Model-based testing
  - Parallel and distributed program testing
  - Formal methods, model checking, test generation, testing tools,
  - Testing for critical embedded systems
  - Testing applied to different paradigms (aspect-oriented programs, SOA, etc.)
- Experimental software engineering
- Methods, techniques and tools for Software Engineering in the educational context
  - Open source software and web application.
  - Software engineering education and training, learning objects and distance education.
  - Educational games
  - Collaborative mobile and distance learning
  - Tools to evaluate programs

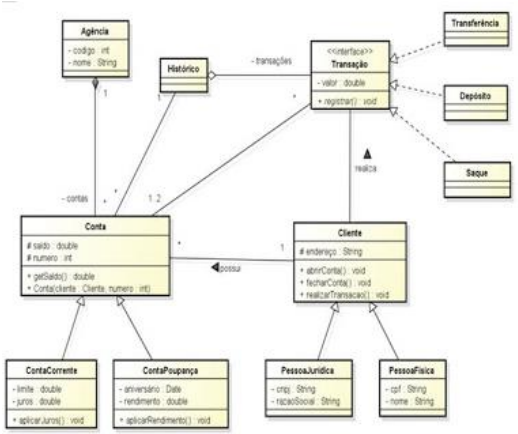
Áreas da engenharia de software segundo a norma ISO/IEC 12207

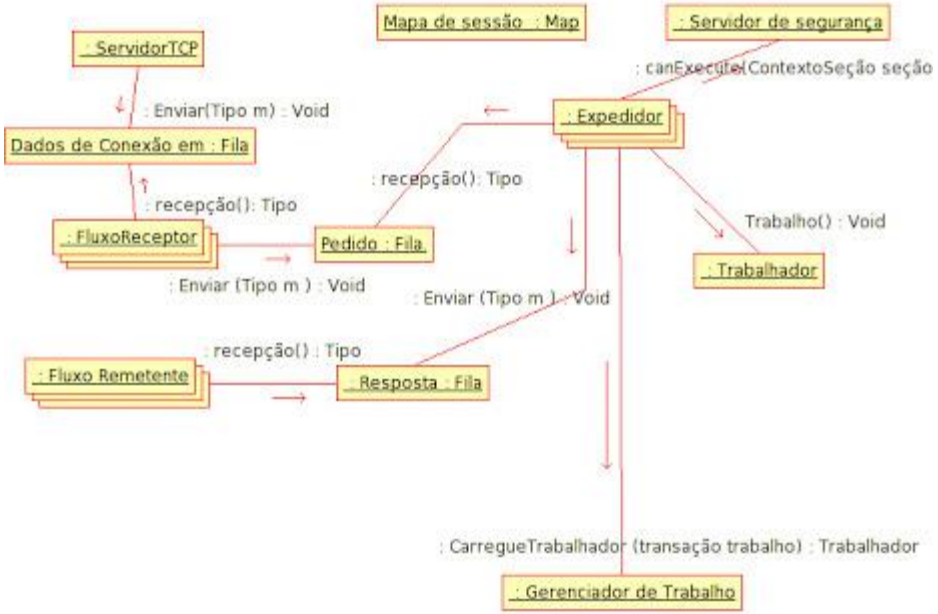


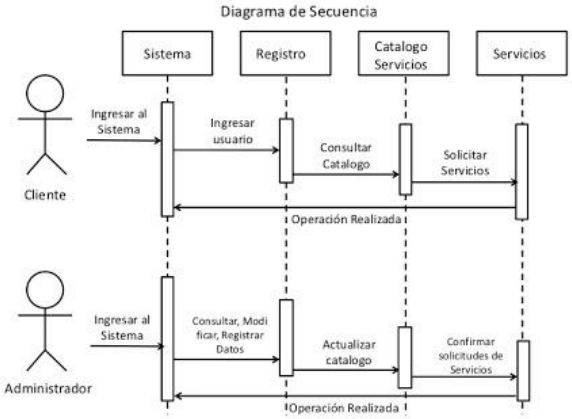
Fim dos anos 80: surgimento da análise orientada a objetos

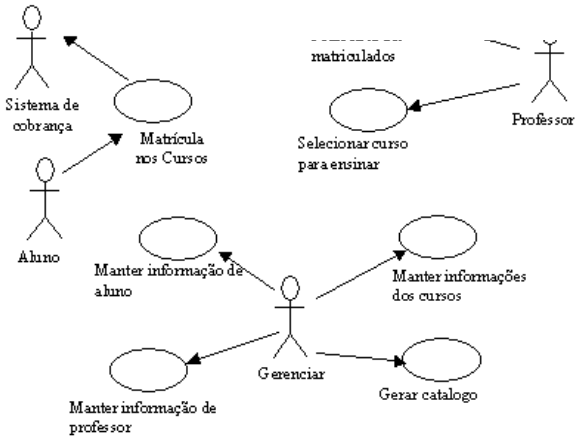
- Após a estabilização das linguagens orientadas a objetos, surgiram vários métodos de análise e projeto OO
  - CRC (*Class Responsability Collaborator*, Beck e Cunningham, **1989**)
  - OOA (*Object Oriented Analysis*, Coad e Yourdon, **1990**)
  - Booch (**1991**)
  - OMT (*Object Modeling Technique*, Rumbaugh, **1991**)
  - Objectory (Jacobson, **1992**)
  - Fusion (Coleman, **1994**)

Fim dos anos 80: surgimento da análise orientada a objetos









1997: a UML (Unified Modeling Language)

- Várias notações para análise orientada a objetos
- Outras tentativas de fusão não deram certo
- UML publicada em 1997 e desde então é a notação mais amplamente utilizada

2001: O manifesto ágil

- **Indivíduos e interações** 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

2001: O manifesto ágil

- Métodos Ágeis:
  - Extreme Programming
  - Crystal Clear
  - Adaptive Software Development
  - SCRUM
  - Feature Driven Development
  - Dynamic Systems Development Method
  - Lean Software Development

Evolução do Software

2000 – atual

- Sistemas Web
- Web *services*
- Software como serviço (cloud computing)
- Aplicativos para smartphone (mobile applications)
- Internet das coisas
- Sistema-de-sistemas

Desafios Atuais da Engenharia de Software

- Sistemas legados
  - Heterogeneidade
    - Entrega dos sistemas

Concluindo...

- Software é de produção complexa...
- Processos de software são complexos...
- Requisitos
  - Custam caro, mas são muito necessários
  - Mudam constantemente
- No silver bullet!