INFO1003 – Eng Sw II

Engenharia de Software

Prof. Marcelo Soares Pimenta

mpimenta@inf.ufrgs.br

slides – arq 1

Programa

1 - Revisão de Fundamentos de Engenharia de Software:

2-Análise e Projeto Orientados a Objetos: Aprofundando a prática de modelagem com UML

3-Teste de Software

4. Gerenciamento de Versões e Configurações e Gerência de Mudanças (de Requisitos)

5. Qualidade de Software e Modelos de Maturidade: CMMI, MPS-BR

7. Reuso de Software: teoria e prática, abordagens, patterns, frameworks, componentes, linhas de produto

8. Métodos Ágeis : Motivação, Características, Exemplos

9.Novas Tendências em Engenharia de Software

Problemas com Software são recentes?

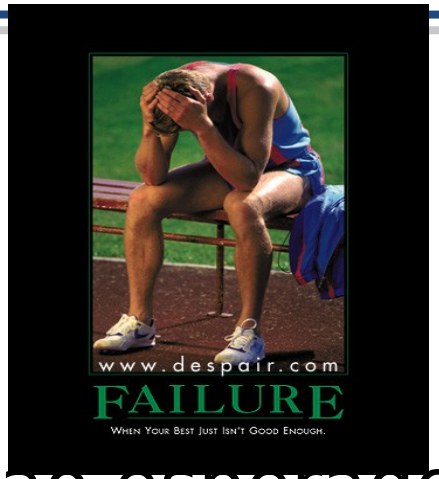
- *What have been the complaints? Typically, they were:*
  - *a) Existing software production is done by amateurs (regardless whether at universities, software houses or manufacturers),*
  - *b) Existing software development is done by tinkering or by the human wave ("million monkey") approach at the manufacturer's,*
  - *c) Existing software is unreliable and needs permanent "maintenance", the word maintenance being misused to denote fallacies which are expected from the very beginning by the producer,*
  - *d) Existing software is messy, lacks transparency, prevents improvement or building on (or at least requires too high a price to be paid for this).*
  - *e) Last, but not least, the common complaint is:*
  - *Existing software comes too late and at higher costs than expected, and does not fulfill the promises made for it.*

F.L. Bauer, Information Processing (IFIP) **1971** Conference Report (Amsterdam: North-Holland Publishing Co, 1972), I, 530-538; extracted from page 530.

Crise de Software (1/2)

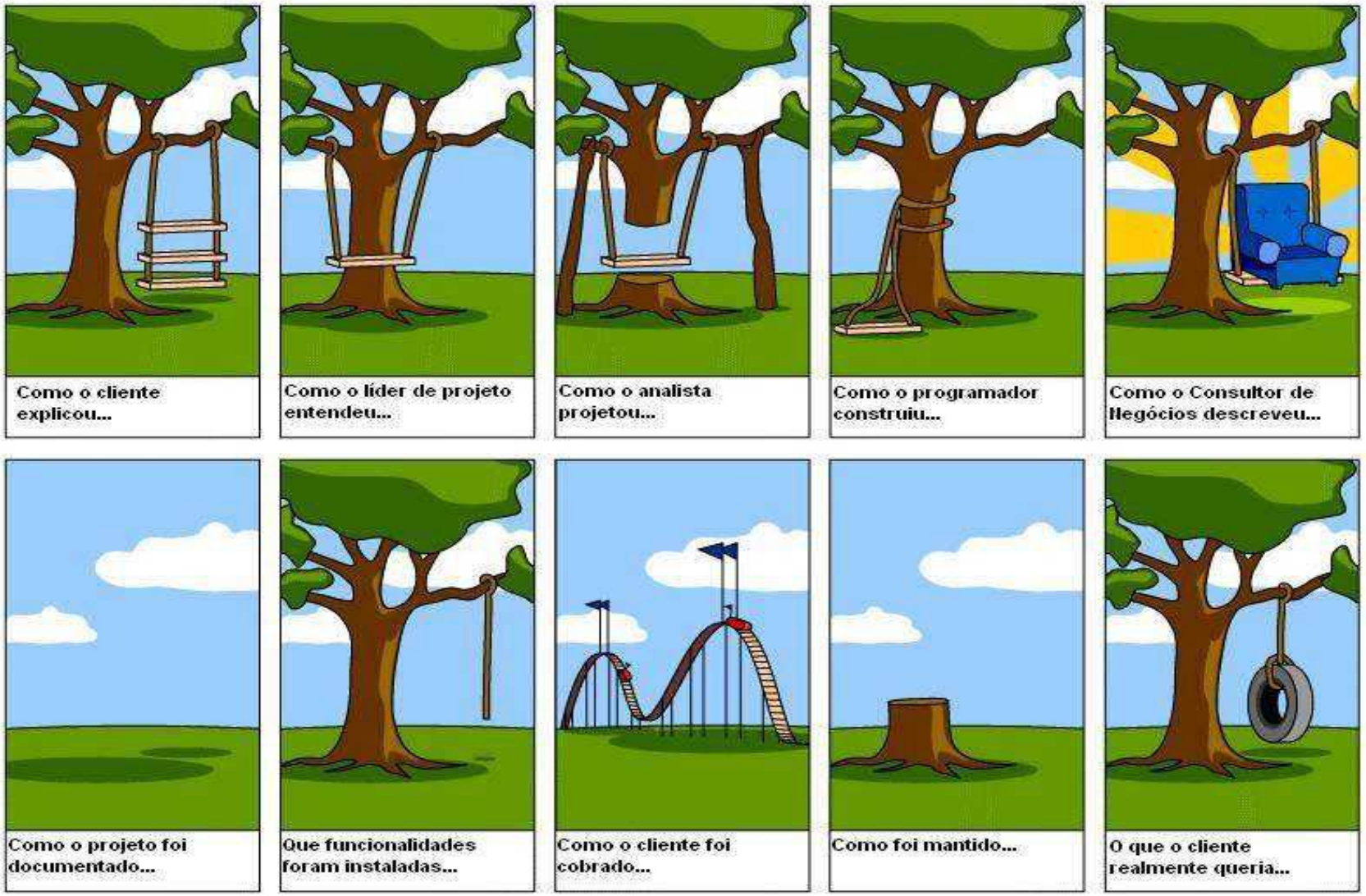
- Crise de Software? Que Crise?
  - Ilusão: Desenvolver Software é "SOFT"
    - Economicamente "soft"
    - Intelectualmente "soft"
    - Operacionalmente "soft"
  - Realidade: Desenvolver Software é "HARD"
    - Satisfazer requisitos (...e usuários)
    - Respeitar orçamento e cronograma
    - Atender restrições de projeto (plataforma, normas, etc)

Crise de Software (2/2)

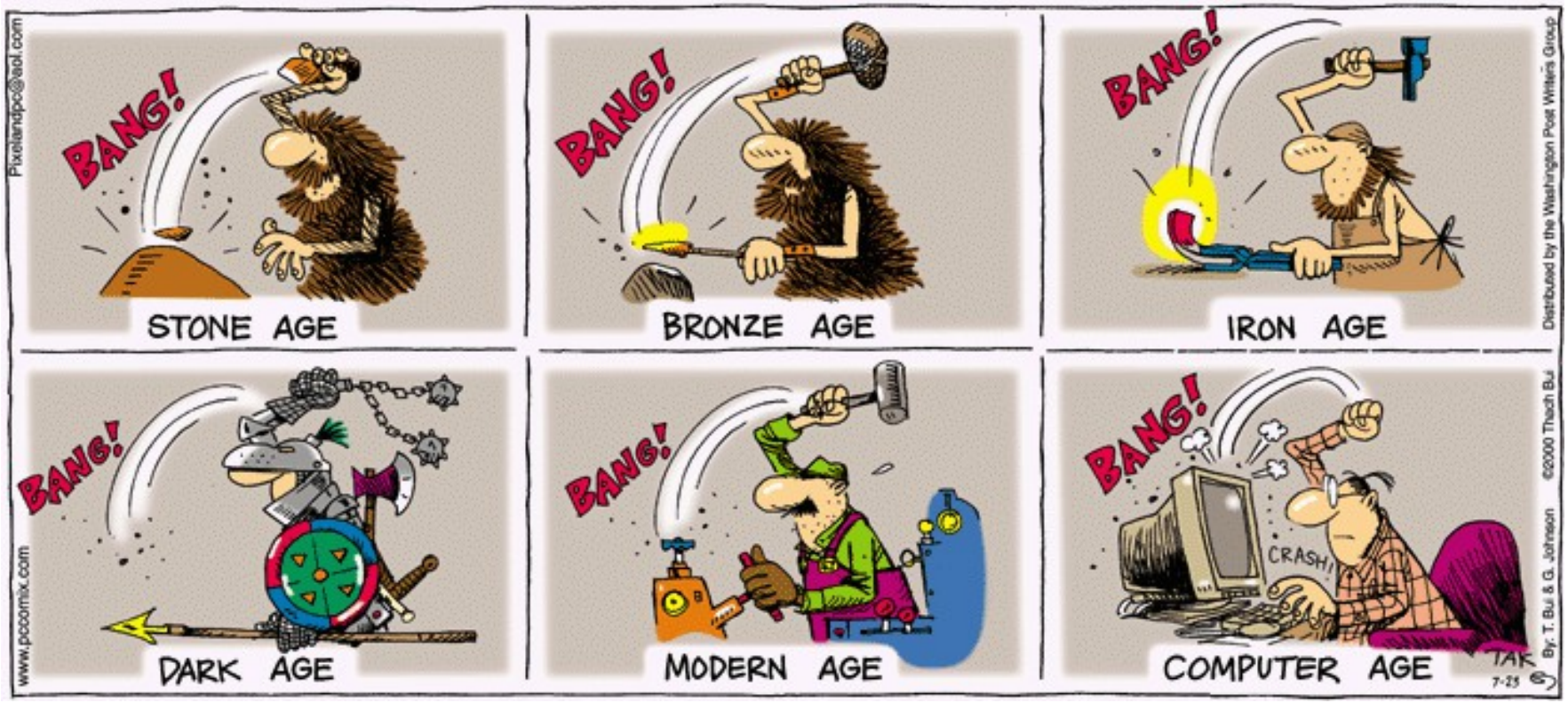


- Crise de Software é
  em verdade Agonia Crônica...
  - Software :
    - Produto de qualidade inferior ao esperado
  - Processo de Software:
    - Processo com baixos graus de satisfação
    - Baixa produtividade dos projetistas de software

Onde está o problema do Software?



Qual o problema das organizações?



Fazer software AINDA é complexo

- Usuários cada mais exigentes: qualidade !!!
- Demandas com complexidade crescente:
  - metodologias tradicionais inadequadas
  - “gap” entre avanço tecnológico e metodológico
- Fazer Software ainda é *knowledge-driven*:
  - Muito baseado em experiência dos desenvolvedores
  - Níveis inadequados de reuso e automação

- Complicadores :
  - Diversidade de plataformas, linguagens, ferramentas...
  - Diversidade de arquiteturas ...

Software: Qual é o problema?

- A demanda por MAIS SOFTWARE e a SOFISTICAÇÃO do software necessário ultrapassaram nossa capacidade de construção ad-hoc...
- Nossa capacidade de manter programas é ameaçada por projetos anteriores ruins.

Situação na Entrega do Software

✓ Excesso de custos	+ 50%
✓ Atraso no cronograma	+ 60%
✓ Não funcionavam	+ 45%
✓ Não foram entregues	+ 29%
✓ Precisaram ser logo modificados	+ 22%
✓ Funcionaram (sucesso)	2%

Fonte: Standish Group, 2008

Motivação para Eng software

1) Atender necessidades dos clientes e reduzir erros e falhas

2) Conseguir gerenciar o desenvolvimento de software (incluindo orçamento e cronograma)

3)Aumentar a qualidade do software produzido

4)Diminuir a dificuldade de manutenção

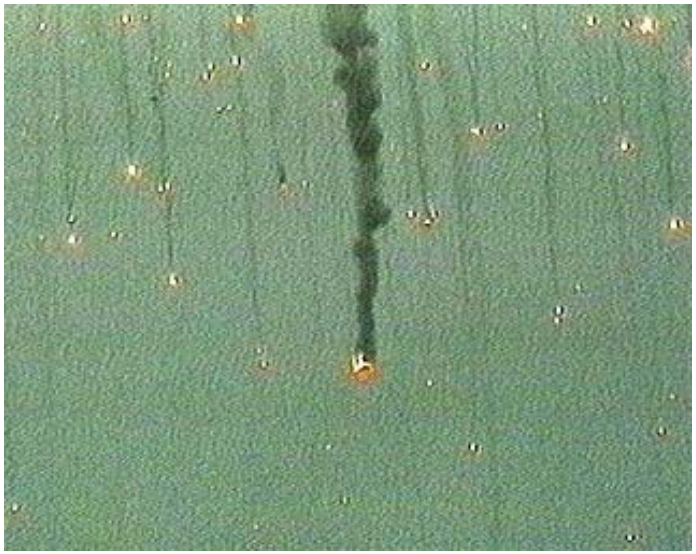
Reduzir erros e falhas?

- Projeto Ariane 5

- Projeto da Agência Espacial Européia que custou:
  - –10 anos.
  - –US$ 8 Bilhões.
- Lançamento em 4/junho/1996
- Garante supremacia européia no espaço.

Ver em http://www.esrin.esa.it/htdocs/esa/ariane/

Erros e Falhas em SW - Resultado



Explosão 40 segundos após a decolagem.

Destruição do foguete e carga avaliada em US$ 500 milhões.

CAUSA: FALHA DE SOFTWARE!

- Shut-down! Ocorrera um *run time error* (out of range, overflow , ou outro) e ambos computadores (principal e back-up) se desligaram.
- Diagnóstico: Um programa que convertia um valor em ponto flutuante para um inteiro de 16 bits recebeu como entrada um valor que estava fora da faixa permitida.

Mais Falhas de Software

- VÔO Air France 447 (Rio-Paris)
  - http://www.airfrance447.com/
  - http://www.spiegel.de/international/world/bild-679980-63086.html
  - Falha em sensor, Piloto Autom toma controle, Pilotos dão reset no piloto, Em 4 minutos o avião caiu no mar
  - **CAUSA:** FALHA DE SOFTWARE ???

- VÔO Oceanic Air 815 (Sydney-LA)
  - http://pt.lostpedia.wikia.com/wiki/Voo_Oceanic_815
  - ;-)

- LISTA DE FALHAS CÉLEBRES:
  - http://en.wikipedia.org/wiki/List_of_software_bugs
  - http://www.sereferences.com/software-failure-list.php

Erros (falhas) e Software

- Primeira premissa: ERRO é fato **cotidiano** e não excepcional
- 3 formas de tratar erros (falhas) de software:

1. Evitar falhas (*fault-avoidance*): especificação, projeto, implementação e manutenção usando métodos sistemáticos e confiáveis , tipicamente baseadas em métodos formais e reuso de componentes de software altamente confiáveis; -> praticável somente para sistemas altamente críticos
2. Eliminar falhas (*fault-elimination*): análise, detecção e correção de erros cometidos durante a desenvolvimento . Aqui incluem-se as atividades verificação, validação e teste.
3. Tolerar falhas (*fault-tolerance*): compensação em tempo real de problemas residuais como mudanças fora da especificação no ambiente operacional, erros de usuário, etc. Geralmente lida com recursos de hardware e/ou software adicionais ou redundantes;

- Devido ao fato de *fault-avoidance* ser economicamente impraticável para a maioria das empresas, e de *fault-tolerance* exigir muitos recursos a tempo de execução, a técnica de eliminação de falhas (*fault-elimination*) geralmente é a adotada pelos desenvolvedores de software.

Erros e Falhas em Software

- *´... software bugs are so common that their cost to the American economy alone is $60 billion a year or about 0.6% of gross domestic product.´ (...)*
- *The same Institute has also calculated that "80% of the software development costs of a typical project are spent on identifying and fixing defects."*
- *Further, the article explained that programming bugs are better found and fixed as early in the development process as possible, since the costs of fixing them escalate as the project progresses.*

[The Economist, June 21st 2003 Technology Quarterly Report]

- Engenharia de Software visa o desenvolvimento de software com minimização de erros.

Custos de Software

- Custos de software dominam os custos de sistemas computadorizados. O custo de software em um PC são (frequentemente) maiores do que o custo do hardware...
- Custos de manutenção são maiores que os custos de desenvolvimento. Para sistemas com uma vida longa, o custo de manutenção pode ser várias vezes o custo do desenvolvimento.
- Engenharia de Software visa o desenvolvimento de software com custo gerenciado.

Desenvolvimento e Manutenção de Software

- Desenvolvimento (30% do esforço):
  - **Início**: quando necessidade do produto é identificada
  - **Fim**: quando teste do produto implantado é concluído e o produto é entregue para a operação/produção
- Manutenção (70 % do esforço)
  - Todas as atividades após a entrega:
    - Aumento da capacidade do produto 60%
    - Adaptação do produto a novos ambientes 20%
    - Correção de erros 20%

Aumentar a qualidade do software

- Qualidade relacionada à conformidade do software com os requisitos:
  - Problema: raramente os requisitos estão completos.
  - Checagem via Verificação
    - Verificação (‘We Build it right **?**’):
      - O que foi especificado e projetado foi construído corretamente?
        - » Checado em relação à especificação e aos requisitos
- Qualidade relacionada à satisfação (das necessidades) do usuário:
  - Problema: usuários diferentes.
  - Checagem via Validação
    - Validação (‘We Build the right thing **?**’):
      - Construímos o que era certo ?
        - » Checado pelo cliente/usuário

Dificuldade de manutenção

- Manutenção é cara e inevitável pois:
  - há muito SW legado (mais de 5 anos de uso)
  - implantação é incompleta (necessita contínuos ajustes)
  - sistemas mudam (refletem ambientes que mudam)

- Como reduzir dificuldade de manutenção ?
  - Melhoria da qualidade do SW produzido
  - **Melhoria do processo de produção de SW**

Requisitos e necessidades

- Definir com clareza os requisitos de um sistema é um grande problema:
  - Compromisso entre requisitos do cliente (usuário) e requisitos do sistema :
    - "Clientes não sabem o que querem".
    - "Clientes mudam de idéia", durante o desenvolvimento de sistemas.
    - Clientes possuem altas expectativas.
    - Alguns requisitos são muito difíceis de especificar (p.ex. Usabilidade)
  - Especificações são muitas vezes incompletas e inconsistentes
  - Mudanças de hardware e software de apoio durante o desenvolvimento.
  - ...

Gerenciar o Processo de Software

- Tarefas:

1) Gerenciar as complexidades do processo de desenvolvimento de software:

Tecnologia, Equipe, Cronograma, Orçamento, Usuários

2) Gerenciar as complexidades dos produtos de software:

Entender, definir, desenvolver, modificar, estender, adaptar os elementos (dados ou atividades) dos sistemas que possuem:

- Multiplicidade de técnicas, notações e linguagens SEM suporte adequado para transformações de uma para outra
- Multiplicidade de plataformas (HW, Sistemas Operacionais) SEM suporte adequado para portabilidade de uma para outra

Gerenciar o Processo de Software

3) Reduzir custos (esforços) e aumentar qualidade e satisfação tanto para os usuários quanto para a equipe de desenvolvimento
    - Implica em soluções de equilíbrio

- Engenharia de Software visa conseguir gerenciar o desenvolvimento de software

Engenharia de Software

- Engenharia = Uso de princípios científicos para uma atividade de projeto e construção
- Engenharia de Software (*n* definições):

  1. /Boehm 76/ "*Aplicação Prática do conhecimento científico para o projeto e a construção de programas computacionais e a documentação necessária à sua operação e manutenção*".

  2. /AFNOR 83/ "*Abordagem sistemática para o desenvolvimento, a operação e a manutenção de software*".

O que é Software?

- SW é um produto ´diferente´
  - **Virtual**: falta de leis e propriedades físicas para SW - visibilidade, massa, volume, cor, odor, etc. - e não degrada com o tempo (torna-se obsoleto funcionalmente mas não desgastado com o uso)
  - **Maleável:** pode ser modificado após pronto
- Dificuldade:
  - NÃO há teoria subjacente (motor elétrico: equações de tensão, potência, corrente, etc): Quais equações a seguir para Software?
  - Todos os artefatos de engenharia existem em um ambiente e para propositos externos, p.ex::pontes, carros, etc.
  - MAS projeto de sw usualmente envolve RE-DESIGN de trabalho ou de organizacao

Engenharia de Software

Programa
Uso Pessoal
Doc pequena
Usuário é o autor
Erro é ´irrelevante´
Sem manutenção

- Programa é artefato
- Desenvolvimento é ´arte´
- Atividade Pessoal (*programming-in-the-small*)

Software
Uso Comercial
Doc rica
Usuários diferenciados
Erro é grave
Muita manutenção

- Software é **produto**
- Desenvolvimento necessita de ´**engenharia**´
- Construção em equipe de SW com múltiplas versões (*programming-in-the-large*)

Engenharia de Software

Engenharia de Software =

- Desenvolver
- Usar
- Reusar
- Integrar

n

★

- Linguagens
- Ferramentas
- Técnicas
- Métodos
- Modelos
- Conceitos
- Princípios
- Equipes de Pessoas

n

Combinação de conhecimentos necessários em **todo o ciclo de vida** do software para a obtenção de **software de qualidade**

Engenharia de Software

- **Questões Básicas da Engenharia de Software**
  - O QUÊ o sistema ou componente faz?
  - COMO o sistema ou componente pode ser composto

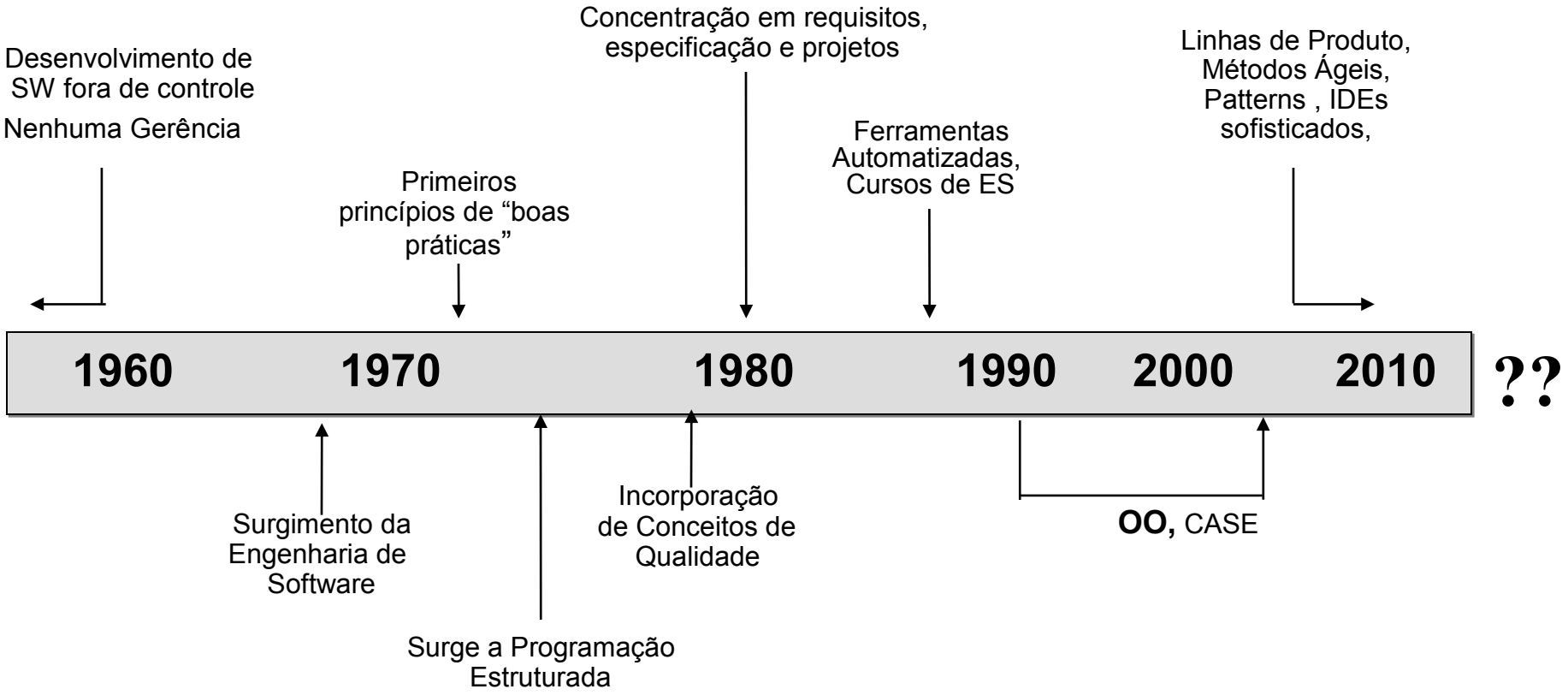
por outros sistemas ou componentes?

  - O QUÊ é um BOM sistema? O que é um sistema BEM construído?

Engenharia de Software

- NA PRÁTICA, enfoque de Engenharia para sistematizar as atividades de:
  - Entender claramente o problema que se quer resolver
  - Desenvolver ferramentas e técnicas para resolvê-lo
  - Gerenciar equipe para resolvê-lo
- Aspectos Tecnológicos e Gerenciais

Evolução da Engenharia de Software



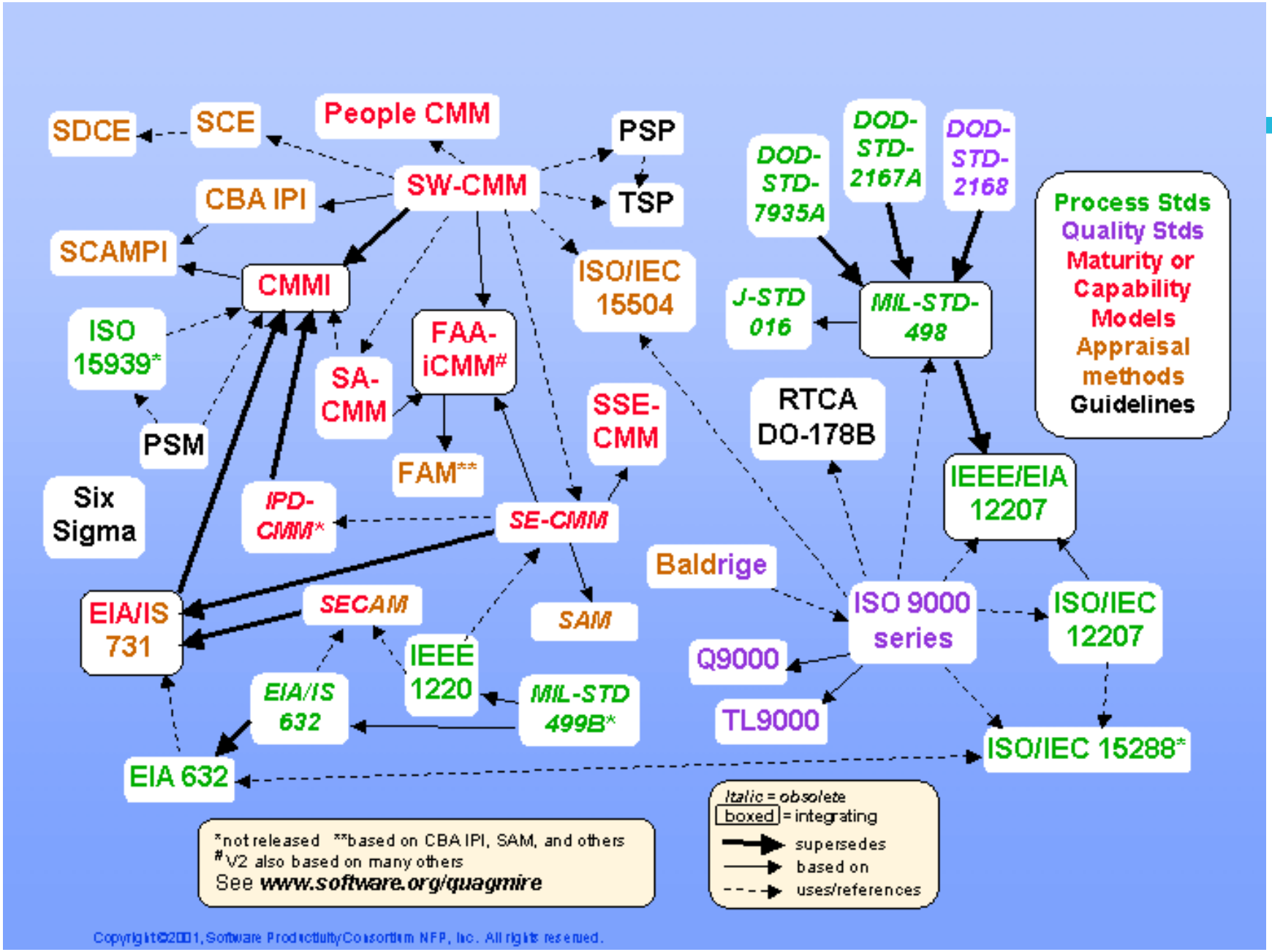
Mas NÃO há uma ´Bala de Prata´!!!

- *Different projects have different needs. Systems have different characteristics, and are built by teams of differing sizes, containing people having differing values and priorities. It cannot be possible to describe the one, best way of producing software.*
- Entao, para conhecer esta diversidade:
  - uma visão panorâmica
  - uma visão em profundidade de alguns tema
  - conciliar teoria e prática

Evolução da Produção de Software

Anos 60-70	**artesanal, waterfall**
Anos 70-80	**Primeiras Metodologias, Aplic. Orientadas a dados**
Anos 80-90	**CMM, Anal. Estruturada**
Anos 90-00	**Fábrica de SW, Outsourcing, PMI, CMMI RUP, UML**

E agora? **Sw livre, Métodos Ágeis, UML 2.0, ...**


SDCE
SCE
People CMM
PSP
TSP
SW-CMM
CBA IPI
SCAMPI
CMMI
ISO/IEC 15504
ISO 15939*
FAA-iCMM#
SA-CMM
SSE-CMM
PSM
FAM**
Six Sigma
IPD-CMM*
SE-CMM
EIA/IS 731
SECAM
SAM
IEEE 1220
EIA/IS 632
MIL-STD 499B*
EIA 632
DOD-STD-7935A
DOD-STD-2167A
DOD-STD-2168
J-STD 016
MIL-STD-498
RTCA DO-178B
IEEE/EIA 12207
Process Stds
Quality Stds
Maturity or Capability Models
Appraisal methods
Guidelines
Baldrige
ISO 9000 series
ISO/IEC 12207
Q9000
TL9000
ISO/IEC 15288*
*not released **based on CBA IPI, SAM, and others
#V2 also based on many others
See www.software.org/quagmire
Italic = obsolete
boxed = integrating
supersedes
based on
uses/references
Copyright ©2001, Software Productivity Consortium NFP, Inc. All rights reserved.

Software engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

FAQs about software engineering

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is a software process model?

FAQs about software engineering

- What are the costs of software engineering?
- What are software engineering methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the attributes of good software?
- What are the key challenges facing software engineering?

What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be
  - Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
  - Bespoke (custom) - developed for a single customer according to their specification.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.
- Examples of process perspectives are
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

What are software engineering methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance.
- Model descriptions
  - Descriptions of graphical models which should be produced;
- Rules
  - Constraints applied to system models;
- Recommendations
  - Advice on good design practice;
- Process guidance
  - What activities to follow.

What is CASE (Computer-Aided Software Engineering)

- Software systems that are intended to provide automated support for software process activities.
- CASE systems are often used for method support.
- Upper-CASE
  - Tools to support the early process activities of requirements and design;
- Lower-CASE
  - Tools to support later activities such as programming, debugging and testing.

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- Maintainability
  - Software must evolve to meet changing needs;
- Dependability
  - Software must be trustworthy;
- Efficiency
  - Software should not make wasteful use of system resources;
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

What are the key challenges facing software engineering?

- Heterogeneity, delivery and trust.
- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- Delivery
  - Developing techniques that lead to faster delivery of software;
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users.

Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation. Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities that are involved in developing software products. Basic activities are software specification, development, validation and evolution.
- Methods are organised ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

Key points

- CASE tools are software systems which are designed to support routine activities in the software process such as editing design diagrams, checking diagram consistency and keeping track of program tests which have been run.

- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.

Leitura Recomendada

- Brooks, F. No Silver Bullet. The Mythical Man-Month - Essays on Software Engineering, (anniversary edition), Addison-Wesley, 1995.

PDF disponível no moodle da disciplina e/ou na HP do professor (conferir)

Segunda Parte

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

1

- *FUNCIONALIDADE*
- **CONFIABILIDADE**
- **USABILIDADE**
- **EFICIÊNCIA**
- **MANUTENIBILIDADE**
- **PORTABILIDADE**

FUNCIONALIDADE

Adequação

PARA AS TAREFAS ESPECIFICADAS

Precisão

RESULTADOS / EFEITOS CORRETOS

Interoperabilidade

INTERAGIR COM SISTEMAS ESPECIFICADOS

Conformidade

A NORMAS, CONVENÇÕES, LEIS, DESCRIÇÕES

Segurança de acesso

EVITAR ACESSO ACIDENTAL OU DELIBERADO

Algumas definições ... ISO 9000:2000

PROCESSO: Um sistema de atividades que usa recursos para transformar entradas em saídas.

PRODUTO: O resultado de um processo.

SISTEMA: Conjunto de elementos inter-relacionados ou interconexos.

CARACTERÍSTICA: Propriedade (coisa) distinguível.

REQUISITO: Necessidade ou expectativa que é declarada, usualmente implícita ou obrigatória.

QUALIDADE: Habilidade de um conjunto de características inerentes de um produto, sistema ou processo para atender plenamente os requisitos dos clientes ou outras partes interessadas.

O que é gestão da qualidade?

ORGANIZAÇÃO: Grupo de pessoas e instalações com um arranjo ordenado de responsabilidades, autoridades e relações.

GESTÃO: Atividades coordenadas para dirigir e controlar uma organização.

GESTÃO DA QUALIDADE: Atividades coordenadas para dirigir e controlar uma organização no que diz respeito à qualidade.

OBJETIVOS DA QUALIDADE: Alguma coisa pensada, ou alvos, relacionados à qualidade.

POLÍTICA DA QUALIDADE: Totalidade das intenções e direção de uma organização relativas à qualidade, formalmente expressas pela alta direção.

SISTEMA DE GESTÃO DA QUALIDADE: Sistema para estabelecer uma política e objetivos da qualidade da qualidade, bem como os métodos para alcançar esses objetivos.

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE

NBR ISO / IEC 9126

- **FUNCIONALIDADE**
- **CONFIABILIDADE**
- **USABILIDADE**
- **EFICIÊNCIA**
- **MANUTENIBILIDADE**
- **PORTABILIDADE**

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

2

- **FUNCIONALIDADE**
- ***CONFIABILIDADE***
- **USABILIDADE**
- **EFICIÊNCIA**
- **MANUTENIBILIDADE**
- **PORTABILIDADE**

CONFIABILIDADE

Maturidade

FREQUÊNCIA DE FALHAS POR DEFEITOS

Tolerância a falhas

CAPACIDADE EM MANTER DESEMPENHO

- FALHAS NO SOFTWARE
- VIOLAÇÃO NAS INTERFACES

Recuperabilidade

DE DADOS E DE DESEMPENHO

TEMPO E ESFORÇO NECESSÁRIOS

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

3

- **FUNCIONALIDADE**
- **CONFIABILIDADE**
- ***USABILIDADE***
- **EFICIÊNCIA**
- **MANUTENIBILIDADE**
- **PORTABILIDADE**

USABILIDADE

Inteligibilidade

ESFORÇO PARA ENTENDER, IDENTIFICAR

Facilidade de Aprendizado

ESFORÇO PARA APRENDER, APLICAR

Operacionalidade

ESFORÇO PARA OPERAR, CONTROLAR

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

4

- **FUNCIONALIDADE**
- **CONFIABILIDADE**
- **USABILIDADE**
- ***EFICIÊNCIA***
- **MANUTENIBILIDADE**
- **PORTABILIDADE**

EFICIÊNCIA

Em relação ao tempo (desempenho)

TEMPO DE RESPOSTA, DE PROCESSAMENTO
VELOCIDADE DE EXECUÇÃO DAS FUNÇÕES

Em relação aos recursos

QUANTIDADE UTILIZADA E
DURAÇÃO DO SEU USO

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

5

- FUNCIONALIDADE
- CONFIABILIDADE
- USABILIDADE
- EFICIÊNCIA
- *MANUTENIBILIDADE*
- PORTABILIDADE

MANUTENIBILIDADE

Analisabilidade

ESFORÇO PARA DIAGNÓSTICO, IDENTIFICAÇÃO DE FALHAS

Modificabilidade

ESFORÇO PARA MODIFICAÇÃO, ADAPTAÇÃO, REMOÇÃO DE DEFEITOS

Estabilidade

RISCO DE EFEITOS INESPERADOS OCASIONADOS POR MODIFICAÇÕES

Testabilidade

ESFORÇO PARA VALIDAÇÃO DAS MODIFICAÇÕES

CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE
NBR ISO / IEC 9126

6

- **FUNCIONALIDADE**
- **CONFIABILIDADE**
- **USABILIDADE**
- **EFICIÊNCIA**
- **MANUTENIBILIDADE**
- ***PORTABILIDADE***

PORTABILIDADE

Adaptabilidade

A OUTROS AMBIENTES POR MEIOS E AÇÕES PRÓPRIAS

Instalabilidade

ESFORÇO PARA A INSTALAÇÃO

Conformidade

ADERÊNCIA A CONVENÇÕES E PADRÕES FORMAIS DE PORTABILIDADE

Capacidade para substituir

ESFORÇO E CAPACIDADE PARA SUBSTITUIR OUTRO SOFTWARE

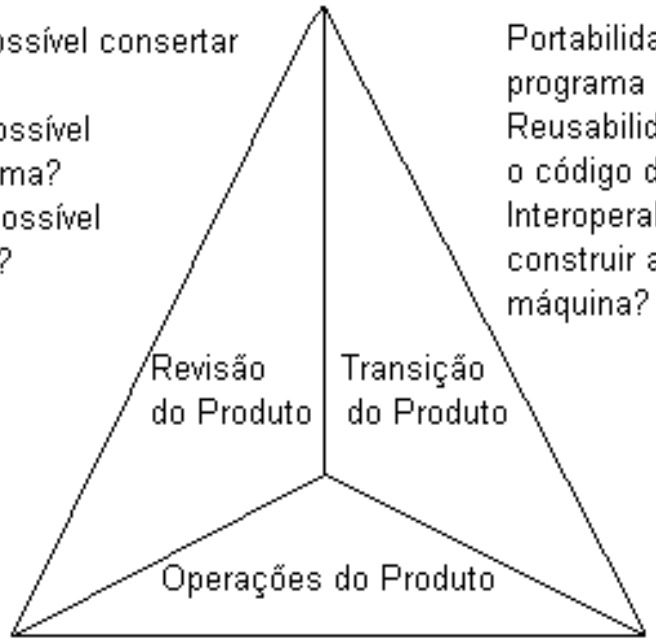
Modelo de McCall - 1977

- Identifica três áreas de trabalho:
  - operação;
  - revisão;
  - transição.

- Identificação de critérios em cada área de trabalho.

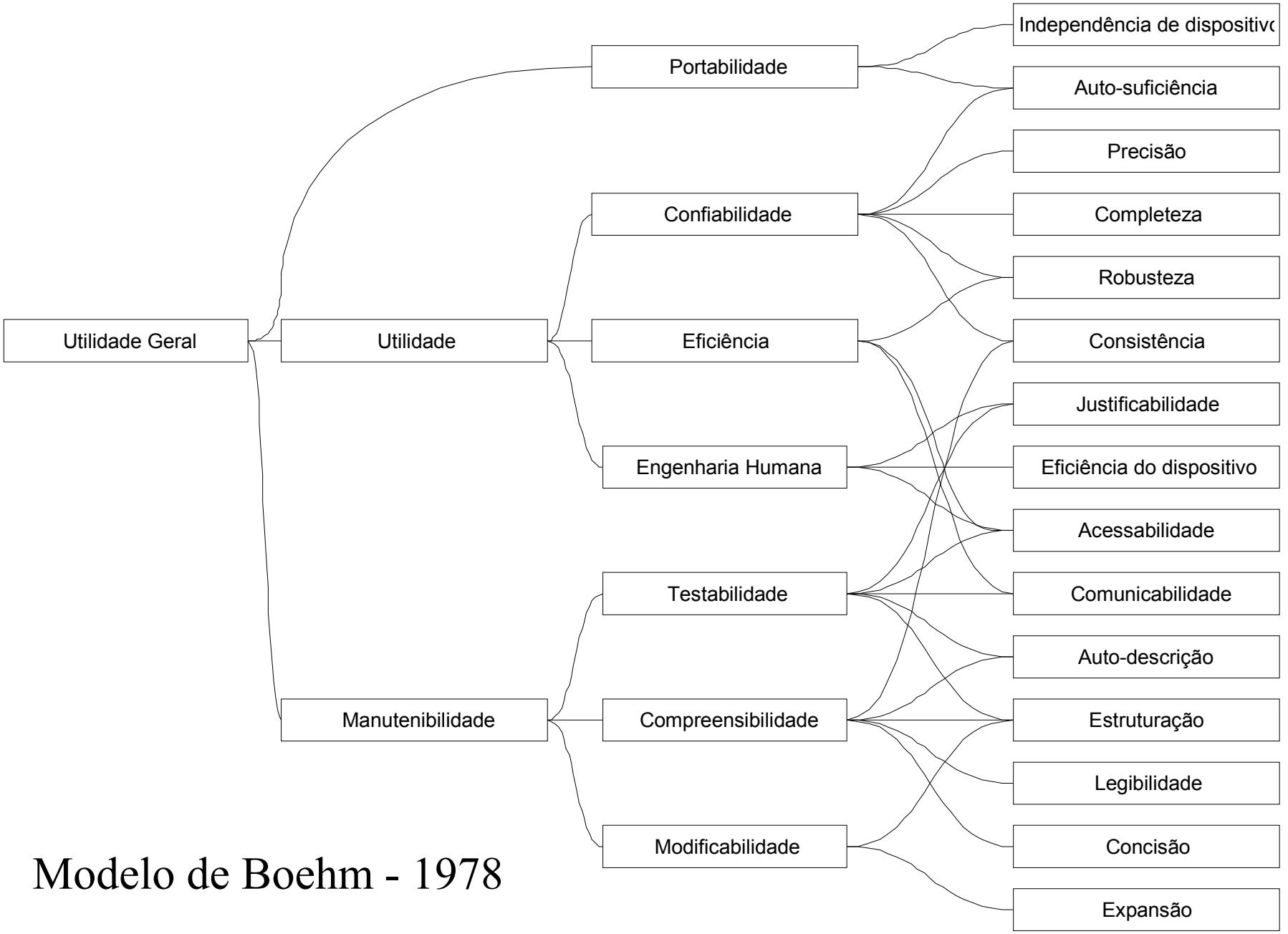
Modelo de McCall - 1977

Manutenção: É possível consertar o programa?
Flexibilidade: É possível modificar o programa?
Testabilidade: É possível testar o programa?



Portabilidade: É possível usar o programa em outra máquina?
Reusabilidade: É possível reutilizar o código do programa?
Interoperabilidade: É possível construir a interface com outra máquina?

Correteza: O programa faz o desejado?
Confiabilidade: O programa faz o desejado de maneira precisa?
Eficiência: O programa executará tão bem quanto possível?
Integridade: O programa é seguro?
Usabilidade: O programa pode ser executado?



Modelo de Boehm - 1978

Modelo de Meyer

- Qualidade Externa
  - visível a usuários
- Qualidade Interna:
  - invisível a usuários mas percebidos (e alcançados) por projetistas e implementadores
  - CHAVE para obtenção da qualidade externa
- Ex de critérios de qualidade Externa:
  - confiabilidade = corretude + robustez,, extensibilidade, reusabilidade, compatibilidade, etc

Programa

- Análise OO de Sistemas Interativos: Modelos, diagramas e atividades
  - Modelagem Comportamental : Casos de uso e seus níveis de abstração
  - Modelagem Conceitual e Modelo de classes
  - Definindo Funcionalidade em Alto Nível
  - Usando Notação UML para Análise OO de Sistemas Interativos: Diagramas básicos e como construí-los