HISTÓRICO DA ENGENHARIA DE SOFTWARE

- **Início** - Workshop realizado na Alemanha em 1968.
- **74** - Diagramas de Warnier-Orr
- **76** - J.S.D - Jackson System Development
- **77** - SADT (Rose)
- **Início dos Anos 80**:
    - Metodologias para análise e projeto:
        - Análise estruturada (De Marco e Gano)
        - Projeto estruturado (Yourdon - Constantine)
- **Fim dos 80**:
    - Orientação a objetos (Myers, Booch)
    - "BOOM" das Ferrmentas Case
- **Futuro próximo**:
    - Integração I.A. x E.S.
    - Automação de todos os métodos do processo de desenvolvimento de software.
    - Assistentes especialistas.

FORÇAS POR TRÁS DA EMERGÊNCIA ENGENHARIA DE SOFTWARE

- Inabilidade para predição de tempo, esforço e custo em desenvolvimento de software.
- Ruim qualidade do software produzido até então.
- Modificação da razão: **Custo de Hardware/Custo de Software**.
- Problemas de manutenção do software.
- Avanços da tecnologia de hardware.
- Avanços em técnicas para software.

- Crescente demanda por novos softwares, maiores e mais complexos.

A CRISE DO SOFTWARE

Diz respeito a uma série de problemas encontrados no desenvolvimento de software, relacionados a **como desenvolvê-lo, mantê-lo e a como suprir a crescente demanda por ele**.

Principais Problemas Encontrados

- Estimativas ruins de **custo e tempo**.
- Baixa produtividade humana.
- Software com qualidade ruim.
- Fracas indicações sobre produtividade, não permitindo a avaliação de novas ferramentas, técnicas e padrões.
- Insatisfação do usuário final.
- Manutenção difícil do software.

CAUSAS DA CRISE DO SOFTWARE

- Capacidade técnica insuficiente das pessoas responsáveis pelo desenvolvimento.
- Pouco treinamento em novas técnicas.
- Resistência imposta pelos responsáveis pelo desenvolvimento à introdução de novas técnicas.
- Falta de processos e métodos.
- Documentação insuficiente.

O PROBLEMA CLÁSSICO DO SOFTWARE

O software é um produto **lógico** e não físico ⇒ custos concentram-se no **desenvolvimento** e não na produção ⇒ a ênfase deve ser dada em técnicas para **gerenciamento**.

MAS AFINAL, O QUE É A ENGENHARIA DE SOFTWARE?

- **Premissa**: É um ramo da Engenharia.
- **IEEE**: "Abordagem sitemática para desenvolvimento, operação e manutenção de software".
- **Baver**: "Uso de sólidos princípios de engenharia de forma a obter-se software economicamente viável, confiável e que funcione".
- **Fairley**: "Disciplina tecnológica e administrativa que trata da manutenção e produção sistemática de produtos de software".

CARACTERÍSTICAS DO SOFTWARE

- Variedade de funções.
- Variedade de implementações
- Evolução
- Visibilidade
- Continuidade
- **Raramente é construído a partir de partes existentes**, como o hardware.

Categorias de Tamanho de um Software

CATEGORIA	Nº DE PROGRAMADORES	DURAÇÃO	TAMANHO (LINHAS)
Trivial	1	1 a 4 sem.	500
Pequeno	1	1 a 2 meses	1K - 2K
Médio	2 a 5	1 a 2 anos	5K - 50K
Grande	5 a 20	2 a 3 anos	50K - 100K
Muito Grande	100 a 2.000	4 a 5 anos	1M
Extr. Grande	2.000 a 5.000	5 a 10 anos	1M - 10M

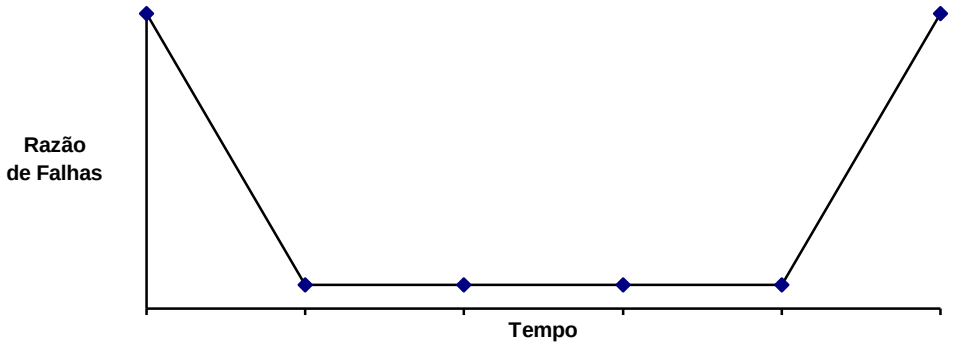


Gráfico: Razão de Falhas para Hardware

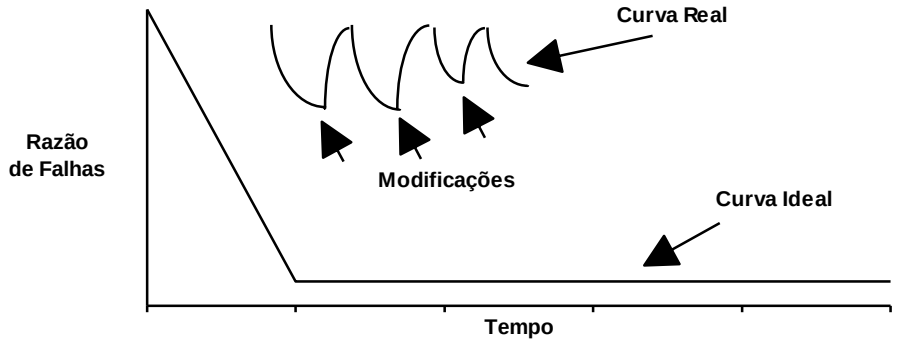


Gráfico: Razão de Falhas para Software

MITOS DO DESENVOLVIMENTO DO SOFTWARE

Mitos da Gerência

- Os programas de hoje são os mesmos de há dez anos. Para que alterar as técnicas de produção?
- Projeto atrasou? Acrescente mais programadores!
- Não preciso de ferramentas modernas quando possuo os mais modernos computadores.

Mitos dos programadores

- Métodos para análise e projeto não funcionam.
- O trabalho está pronto quando o programa funcionar.
- O único indicador do sucesso de um programa é estar o mesmo funcionando.
- A qualidade de um programa só pode ser avaliada quando ele estiver "rodando".

ENGENHARIA DE SOFTWARE X ENGENHARIA

Diferenças

- Software é um produto lógico
- Continuidade
- Visibilidade
- Software não está sujeito a leis físicas
- Software é visualizado como um processo de produção de um serviço. Isto gera dúvidas:
  - Pode ser patenteado?
  - Pagar ICMS ou não?
- As interfaces inter-módulos não são tão visíveis no produto de engenharia.

Semelhanças

- Ciclo de vida.
- Gerenciamento.
- Organização.

O IMPACTO DE ERROS DE SOFTWARE

Erros normalmente resultam em dois custos:

- O dano deles resultante.
- O custo para correção.

(Gráfico 3)

Por quê?

- Testes tornan-se mais complexos.
- Modificações envolvem mais pessoas.
- Repetição de testes de fases anteriores é mais difícil.
- Mudanças acarretam outras modificações em fases anteriores.
- Responsáveis por fases anteriores podem não estar presentes.

ESTATÍSTICAS

- Linhas de código produzidas no mundo:

  100 Bilhões

  70 Bilhões ⇒ COBOL

- Guerra nas Estrelas:

  100 M L.O.C. ⇒ 30 a 50.000 **ERROS**!!!

- Space Shuttle:

  500.000 L.O.C. a bordo.

  1.700.00 L.O.C. em Terra.

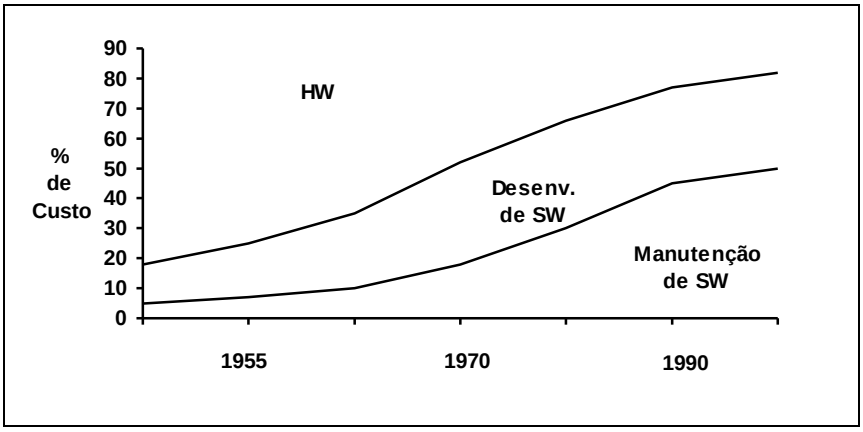
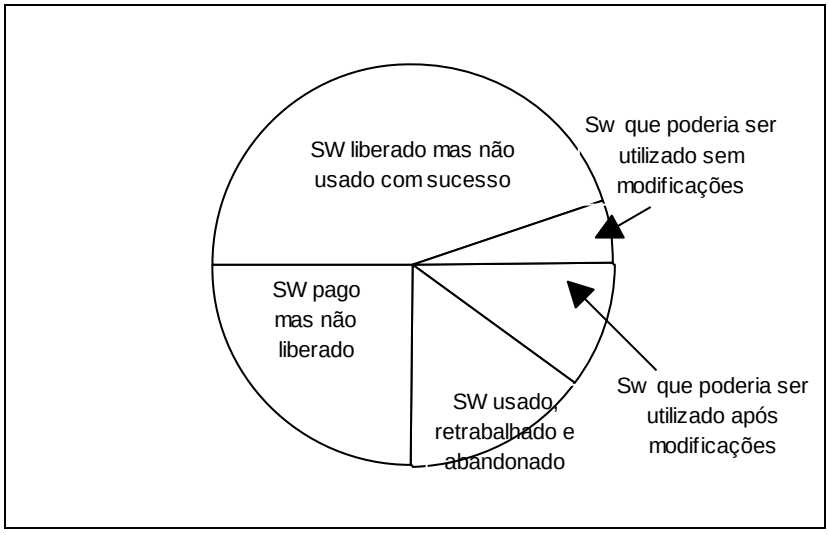
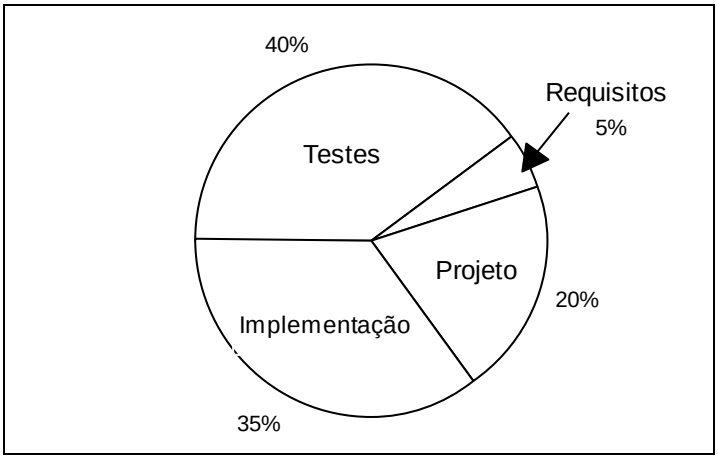
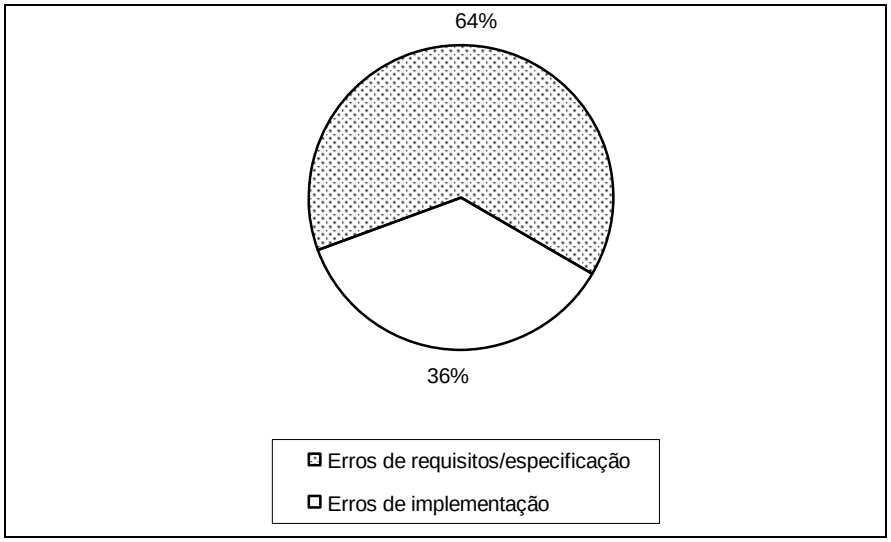


Gráfico: Tendência do Custo Hardware/Software




40%
Requisitos
5%
Testes
Projeto
20%
Implementação
35%


64%
36%
Erros de requisitos/especificação
Erros de implementação

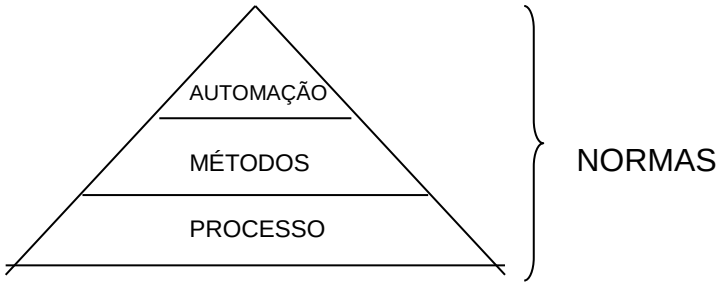
AMBIENTES PARA ENGENHARIA DE SOFTWARE

Definição

"Processo, Métodos e Automação requeridos para produzir um sistema de software."

Objetivo

Prover um contexto para uma evolução racional, ordenada e bem gerenciada do software.



Processo:

Cadeia de eventos que descrevem:

- ATIVIDADES - Tarefas de desenvolvimento e gerenciamento
- PRODUTOS - Saídas das tarefas

Métodos:

- Tudo aquilo necessário à DEFINIÇÃO, DESCRIÇÃO, ASTRAÇÃO, MODIFICAÇÃO, REFINAMENTO E DOCUMENTAÇÃO do software.
- Devem suportar as ATIVIDADES definidas no processo.

Automação:

Uso do computador para implementação dos métodos (CASE)

O PROCESSO

Características de um bom processo:

- Generabilidade
- Adequabilidade à automação
- Alto poder descritivo
- Possuir produtos capazes de documentar convenientemente o software
- Possibilitar integração entre suas diversas fases

OS MÉTODOS

- Devem implementar as atividades e possibilitar a criação dos produtos do PROCESSO
- Devem ser integrados: TRACEABILIDADE entre fases

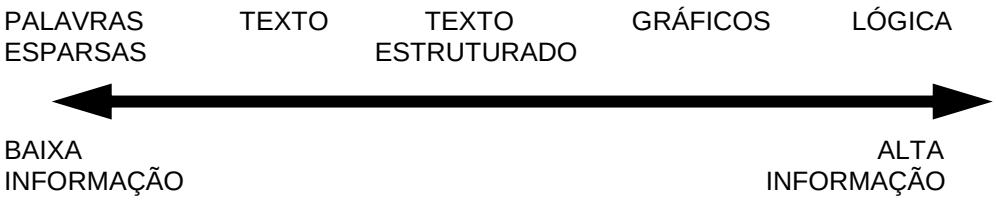
Propriedades de um método:

1. PONTO DE VISTA
   É a perspectiva sob a qual o método encara o problema

   Por exemplo:
   - Visualização de relacionamento entre dados (PONTO DE VISTA INFORMACIONAL)
   - Visualização do fluxo dos dados e das transformações de dados (PONTO DE VISTA FUNCIONAL)
   - Visualização do sequenciamento e da transformação dos dados (PONTO DE VISTA COMPORTAMENTAL)

2. MEIO
   É o recurso que o método utiliza para implementar as atividades.

   Por exemplo:



3. DOMÍNIO
   É o escopo de aplicação do método, determinado pelo produto a ser desenvolvido e pela fase onde reside este produto.

   Por exemplo, a análise estruturada não serve para emissão de um projeto de software (FASE), nem para especificar sistemas tempo-real (PRODUTO).

4. ORIENTAÇÃO
   Regras a serem seguidas para uso correto do método:
   - Que decisões tomar?
   - Como tomar decisões?
   - Em que ordem tomar as decisões?
   - Quando começar?
   - Quando parar?

Princípios básicos utilizados em um bom método:

- Modularidade
- Boa capacidade de abstração
- Descrever a informação com uniformidade
- Ser capaz de representar toda e qualquer informação
- Ser capaz de orientar na utilização das informações realmente necessárias para descrever o sistema
- Ter capacidade de agrupar estruturas semanalmente juntas

A AUTOMAÇÃO

Significado

Prover um ambiente automatizado de suporte a métodos.

Observações Importantes:

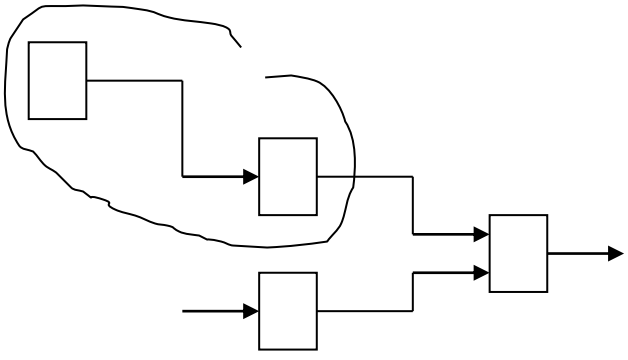
- O método a ser automatizado deve interagir com outros métodos, de outras fases.
- Jamais automatizar um ambiente sem antes possuir um processo e um conjunto de métodos perfeitamente definidos.

Benefícios

- Reduz trabalho no uso dos métodos
  $\Rightarrow$ Aumenta produtividade e reduz custos
- Alguns métodos só são viáveis quando automatizados, pois o volume de informação e processamento necessários são impossíveis de serem manualmente manipulados
- Aumento da criatividade do analista, pois o mesmo pode se concentrar na tarefa ao invés de se concentrar no método

Como e o quê automatizar?

✓ SIM



- ✓ NÃO (Problemas de interface inter-módulos)