



# Software e Engenharia de Software

## ENGENHARIA DE SOFTWARE - PRESSMAN

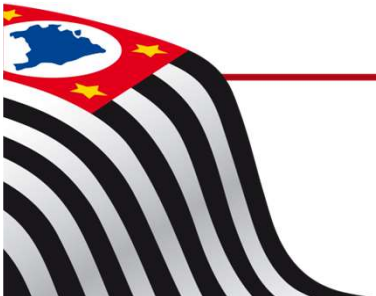


Prof. Antonio Guardado  
Fatec Ipiranga



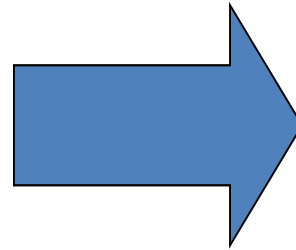
# A importância do Software

- Durante as 3 primeiras décadas da era do computador, o principal desafio era desenvolver um **HARDWARE** de baixo custo e alto desempenho.
- Hoje o desafio é melhorar a qualidade (e reduzir os custos) das soluções baseadas em **SOFTWARE!**

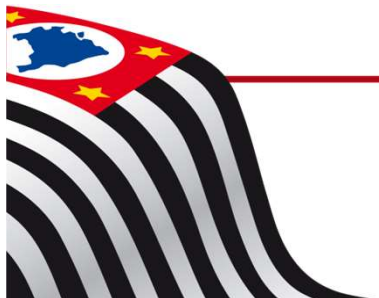


# O Software é o que faz a diferença!!!

- *Completeza* da informação
- *user-friendlynness*
- *web-enhanced*
- inteligência
- funcionalidade
- compatibilidade
- suporte



Tornam um produto  
melhor que outro



# Software

## 1- INSTRUÇÕES

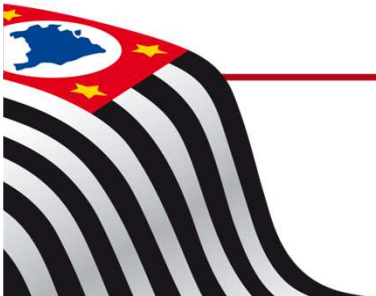
que quando executadas produzem a função e o desempenho desejados

## 2 - ESTRUTURAS DE DADOS

que possibilitam que os programas manipulem adequadamente a informação

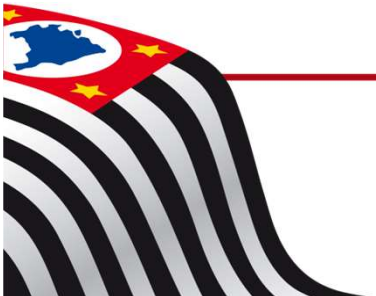
## 3 - DOCUMENTOS

que descrevem a operação e o uso dos programas

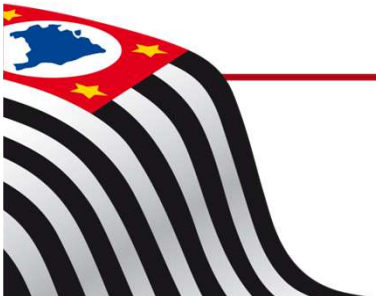
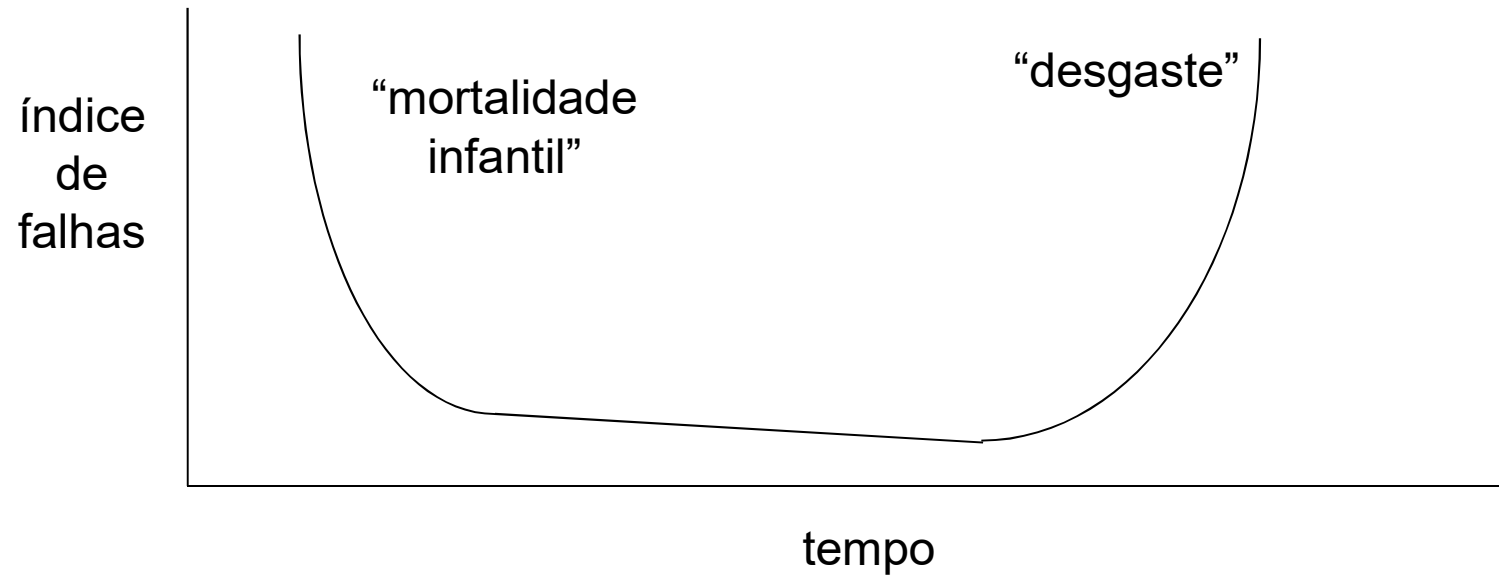


# Características do Software

- 1- desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico
- 2- não se desgasta mas se deteriora
- 3- a maioria é feita sob medida em vez de ser montada a partir de componentes existentes

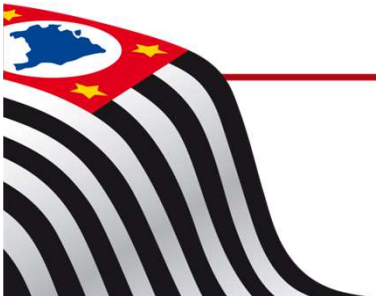
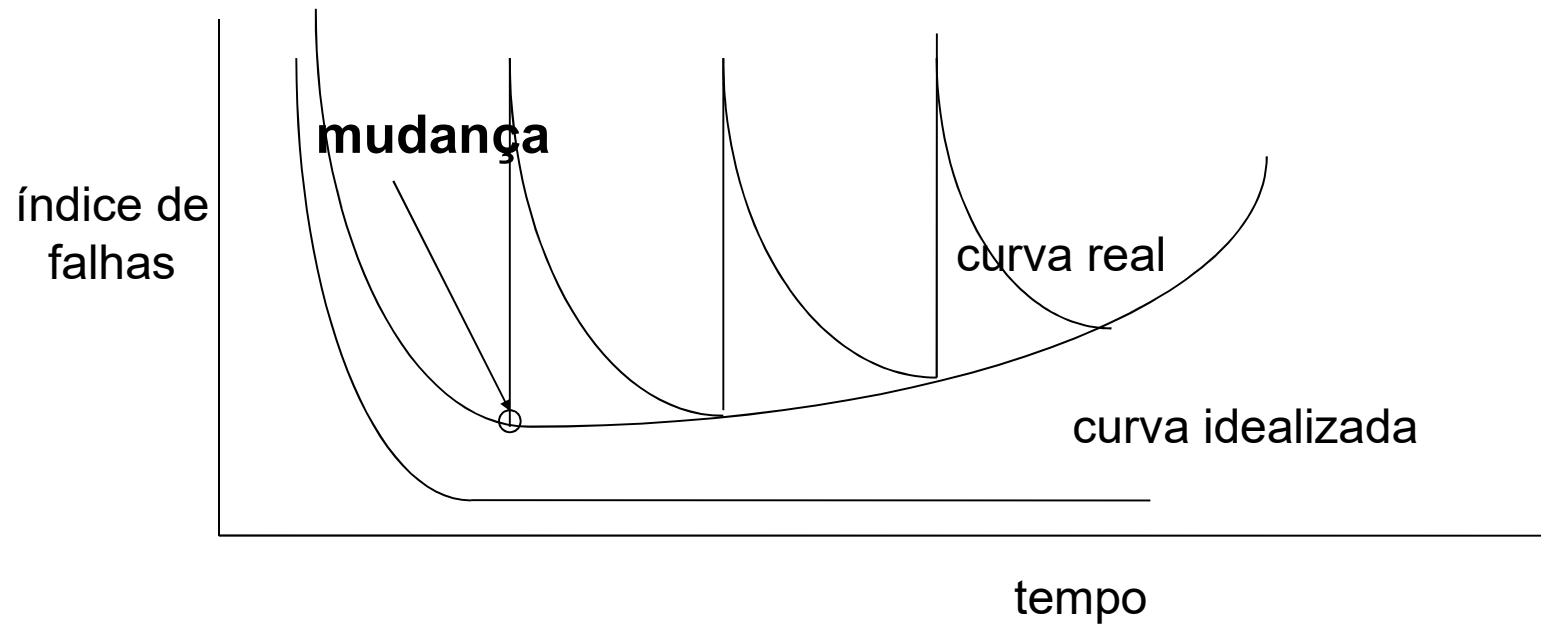


# Curva de falhas para o hardware



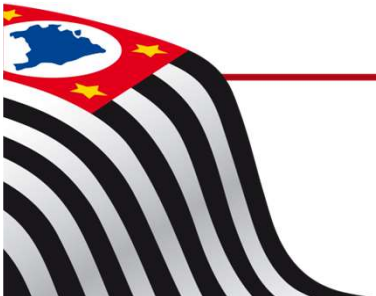


# Curva de falhas do software



# Aplicações do software

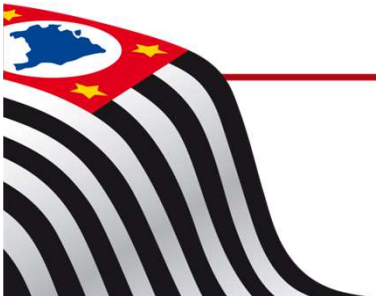
- ☐ BÁSICO *coleção de programas escritos para dar apoio a outros programas*
- ☐ DE TEMPO REAL *software que monitora, analisa e controla eventos do mundo real*
- ☐ COMERCIAL *sistemas de operações comerciais e tomadas de decisões administrativas*
- ☐ CIENTÍFICO E DE ENGENHARIA *caracterizado por algoritmos de processamento de números*





# Aplicações do software

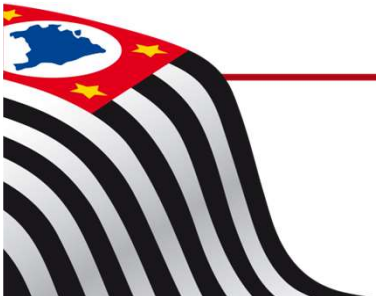
- ☐ EMBUTIDO ou EMBARCADO *usado para controlar produtos e sistemas para os mercados industriais e de consumo*
- ☐ DE COMPUTADOR PESSOAL *envolve processamento de textos, planilhas eletrônicas, diversões, etc.*
- ☐ DE INTELIGÊNCIA ARTIFICIAL *faz uso de algoritmos não numéricos para resolver problemas que não sejam favoráveis à computação ou à análise direta*



# Evolução do software

(1950 - 1965)

- 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



# Evolução do software

(1965 - 1975)

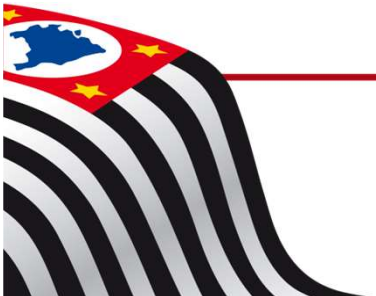
- Multiprogramação e sistemas multiusuários
  - Técnicas interativas
  - Sistemas de tempo real
  - 1ª geração de SGBD's
  - Produto de software - *software houses*
  - Bibliotecas de Software
  - Cresce quantidade de sistemas baseado em computador
  - Manutenção quase impossível
- SOFTWARE** ➔ **CRISE DE**



# Evolução do software

(1975 - *hoje*)

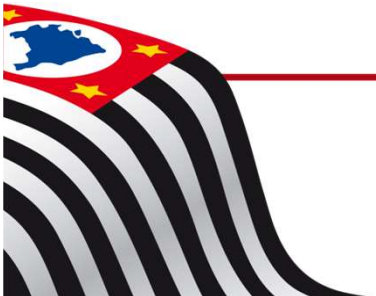
- Sistemas distribuídos
- Redes locais e globais
- Uso generalizado de microprocessadores - produtos inteligentes
- Hardware de baixo custo
- Impacto de consumo



# Evolução do software

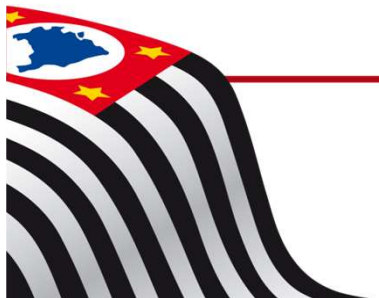
(Quarta era do software de computador)

- Tecnologias orientadas o objetos
- Sistemas especialistas e software de inteligência artificial usados na prática
- Software de rede neural artificial
- Computação Paralela



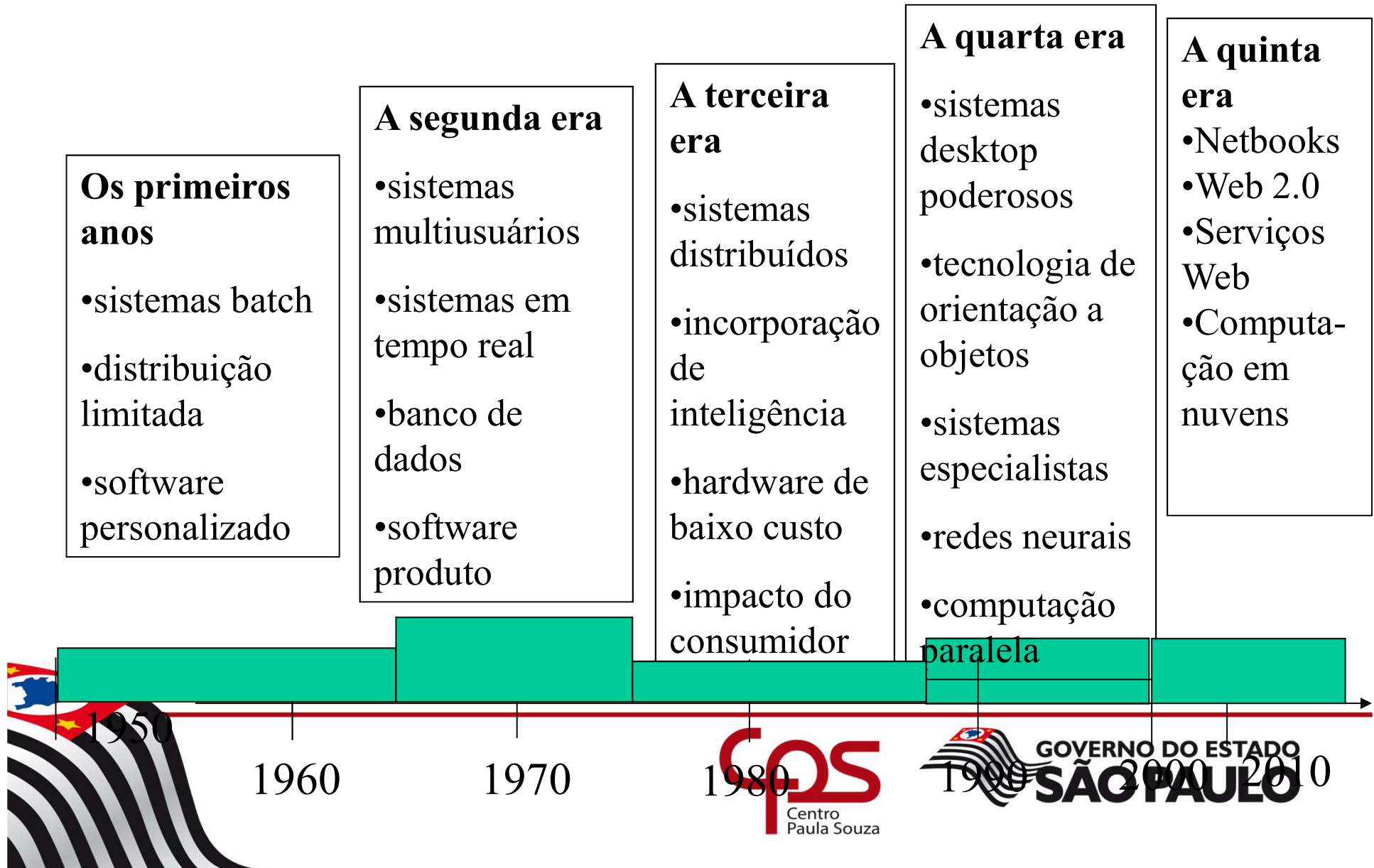
# Uma Crise no horizonte

- A indústria de Software tem tido uma “crise” que a acompanha há quase 30 anos:
  - Aflição Crônica != **Crise**
- Problemas não se limitam ao software que não funciona adequadamente, mas abrange:
  - desenvolvimento, testes, manutenção, suprimento, etc.



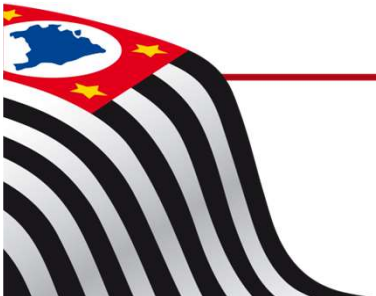


# A Evolução do Software



# Problemas : Therac-25

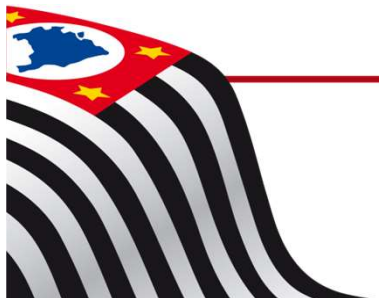
- Equipamento de Radioterapia.
- Entre 1985 e 1987 se envolveu em 6 acidentes, causando mortes por overdoses de radiação.
- Software foi adaptado de uma antecessora, Therac-6:
  - falhas por falta de testes integrados
  - falta de documentação



# Denver International Airport



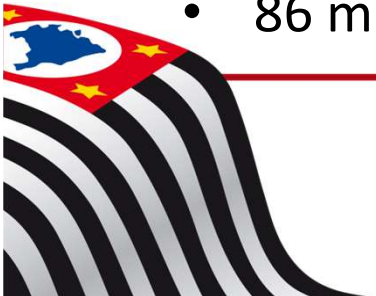
- Custo do projeto: US\$ 4.9 bilhões
  - 100 mil passageiros por dia
  - 1,200 vôos
  - 53 milhas quadradas
  - 94 portões de embarque e desembarque
  - 6 pistas de pouso / decolagem



# Denver International Airport

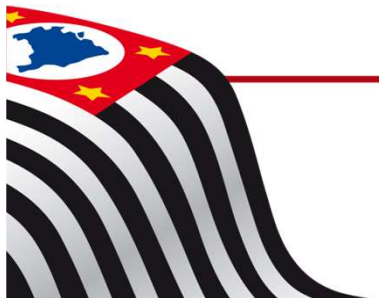


- Erros no sistema automático de transporte de bagagens (*misloaded, misrouted, jammed*):
  - Atraso na abertura do aeroporto com custo total estimado em US\$360 Milhões
- 86 milhões para consertar o sistema



# 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.



# Crise de software

Refere-se a um conjunto de problemas encontrados no desenvolvimento de software:

- 1- *As estimativas de prazo e de custo frequentemente são imprecisas*

“Não dedicamos tempo para coletar dados sobre o processo de desenvolvimento de software”

“Sem nenhuma indicação sólida de produtividade, não podemos avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões”

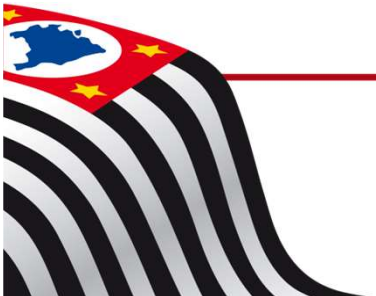




# Crise de software

2- A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços

“Os projetos de desenvolvimento de software normalmente são efetuados apenas com um vago indício das exigências do cliente”



# Crise de software

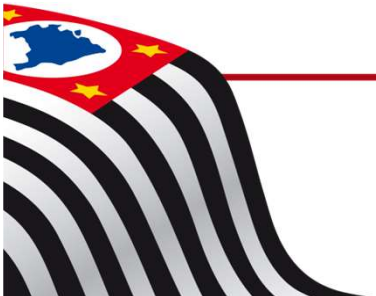
## 3- A qualidade de software às vezes é menos que adequada

Só recentemente começaram a surgir conceitos quantitativos sólidos de garantia de qualidade de software

## 4- O software existente é muito difícil de manter

A tarefa de manutenção devora o orçamento destinado ao software

A facilidade de manutenção não foi enfatizada como um critério importante



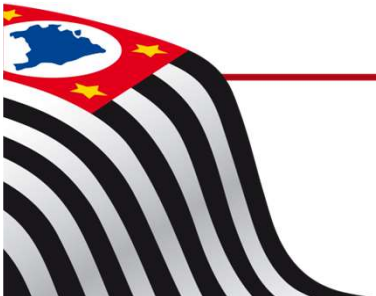
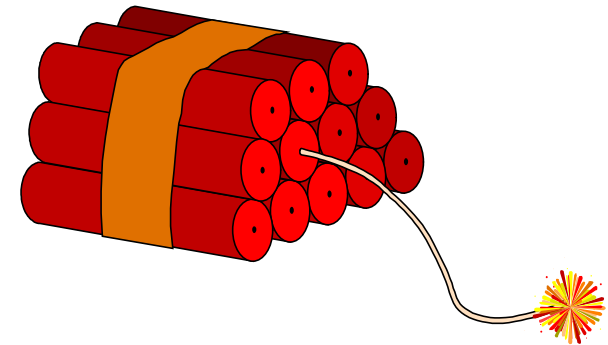
# Crise de software

✓ estimativas de prazo e de custo ↑

✓ produtividade das pessoas ↓

✓ qualidade de software ↓

✓ software difícil de manter ↑

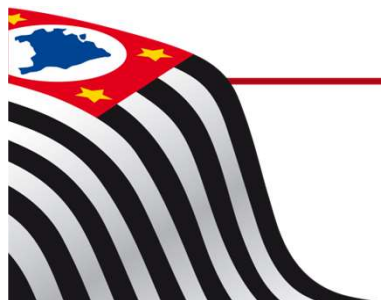


# Causas dos problemas associados à **crise de software**

## 1- PRÓPRIO CARÁTER DO SOFTWARE

O software é um elemento de sistema lógico e não físico. Conseqüentemente o sucesso é medido pela qualidade de uma única entidade e não pela qualidade de muitas entidades manufaturadas

O software não se desgasta, mas se deteriora



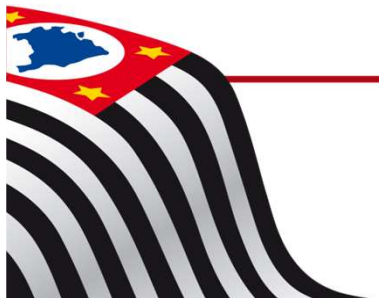
# Causas dos problemas associados à **crise de software**

## 2- FALHAS DAS PESSOAS RESPONSÁVEIS PELO DESENVOLVIMENTO DE SOFTWARE

Gerentes sem nenhum *background* em software

Os profissionais da área de software têm recebido pouco treinamento formal em novas técnicas para o desenvolvimento de software

Resistência a mudanças.

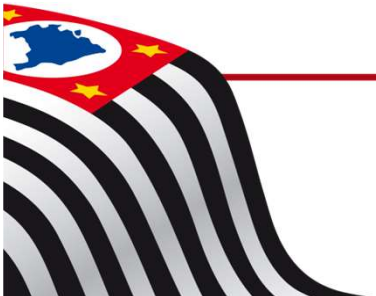


# Causas dos problemas associados à **crise** **de software**

## 3- MITOS DO SOFTWARE

Propagaram desinformação e confusão

- ▶ *administrativos*
- ▶ *cliente*
- ▶ *profissional*





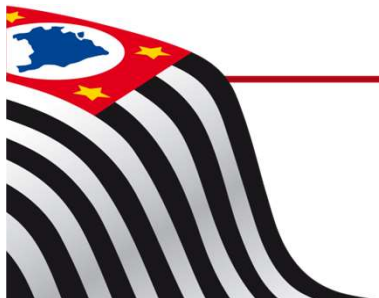
# Mitos do software (ADMINISTRATIVOS)

Mito: 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?*

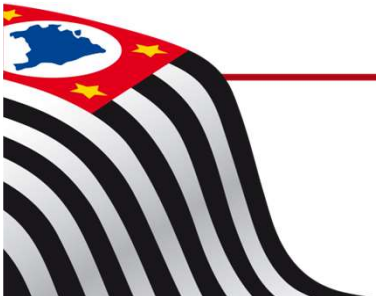
*Ele reflete a prática moderna de desenvolvimento de software? Ele é completo?*



# Mitos do software (ADMINISTRATIVOS)

Mito: Meu pessoal tem ferramentas de desenvolvimento de software de última geração; afinal lhes compramos os mais novos computadores.

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

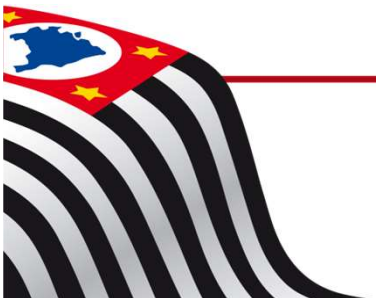


# Mitos do software (ADMINISTRATIVOS)

Mito: 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.*

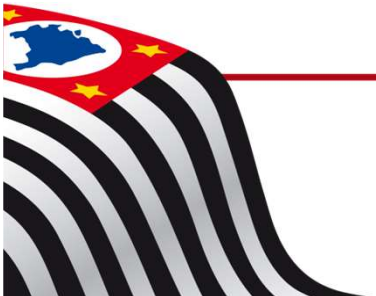
*Pessoas podem ser acrescentadas, mas somente de uma forma planejada.*



# Mitos do software (CLIENTE)

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

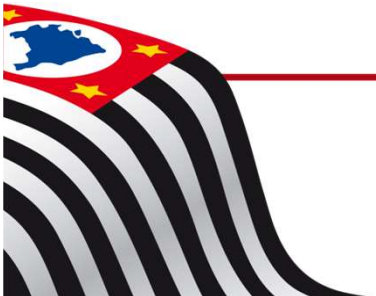
Realidade: *Uma definição inicial ruim é a principal causa de fracassos dos esforços de desenvolvimento de software. É fundamental uma descrição formal e detalhada do domínio da informação, função, desempenho, interfaces, restrições de projeto e critérios de validação.*



# Mitos do software (CLIENTE)

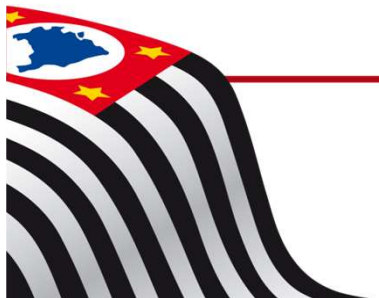
Mito: Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível.

Realidade: *Uma mudança, quando solicitada tardiamente num projeto, pode ser maior do que a ordem de magnitude mais dispendiosa da mesma mudança solicitada nas fases iniciais.*



## Magnitude das mudanças

FASES	CUSTO DE MANUTENÇÃO
DEFINIÇÃO	1 x
DESENVOLVIMENTO	1.5 - 6x
MANUTENÇÃO	60 - 100x

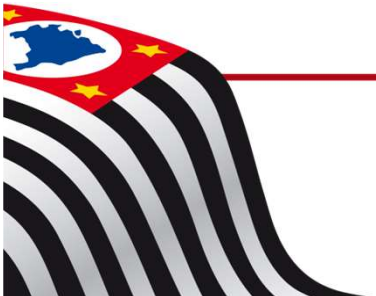




# Mitos do software (PROFISSIONAL)

Mito: Assim que escrevermos o programa e o colocarmos em funcionamento nosso trabalho estará completo.

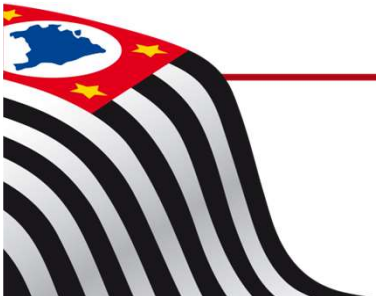
Realidade: Os dados da indústria indicam que entre 50 e 70% de todo esforço gasto num programa serão despendidos depois que ele for entregue pela primeira vez ao cliente.



# Mitos do software (PROFISSIONAL)

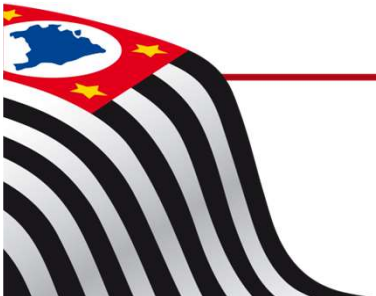
Mito: Enquanto não tiver o programa "funcionando", eu não terei realmente nenhuma maneira de avaliar sua qualidade.

Realidade: *Um programa funcionando é somente uma parte de uma Configuração de Software que inclui todos os itens de informação produzidos durante a construção e manutenção do software.*



# Engenharia de Software

- “Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais”
- É METODOLOGIA! envolve princípios filosóficos que guiam uma gama de métodos que utilizam ferramentas e práticas diferenciadas para realizar algo.



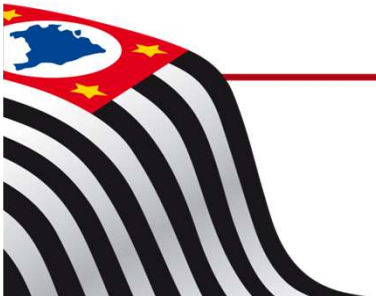
# Engenharia de Software: Elementos Fundamentais

□ **Método ou técnica:** Proporcionam os detalhes de “como fazer” para construir o software.

■ Tarefas:

- Planejamento e estimativa do projeto
- Análise dos requisitos do software
- Projeto da estrutura dos dados
- Arquitetura dos programas
- Codificação
- Teste e manutenção

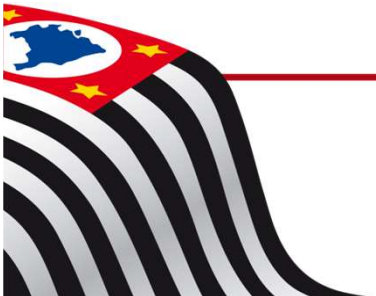
Ex.: um chefe de cozinha prepara um molho combinando ingredientes em uma ordem e momentos específicos.



# Engenharia de Software: Elementos Fundamentais

Ex.:batedeira elétrica

- **Ferramenta:** instrumento ou apoio automatizado para realizar alguma coisa.
  - ⇒ Existem atualmente ferramentas para sustentar cada um dos métodos
  - ⇒ Quando as ferramentas são integradas é estabelecido um sistema de suporte ao desenvolvimento de software chamado *CASE - Computer Aided Software Engineering*



# Engenharia de Software: Elementos Fundamentais

**Procedimento:** constituem o elo de ligação que mantêm juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador.

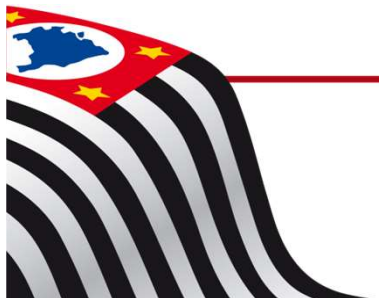
Ex.: plano de testes.


- Produtos que se exige que sejam entregues
  - Controles que ajudam assegurar a qualidade e coordenar as alterações
  - Marcos de referência que possibilitam administrar o progresso do software.
- Os procedimentos definem a seqüência em que os métodos serão aplicados.



# O que é processo, método ou ferramenta?

1. Coloque em uma panela funda o leite condensado, a margarina e o chocolate em pó.
2. Cozinhe [no fogão] em fogo médio e mexa sem parar com uma colher de pau.
3. Cozinhe até que o brigadeiro comece a desgrudar da panela.
4. Deixe esfriar bem, então unte as mãos com margarina, faça as bolinhas e envolva-as em chocolate granulado

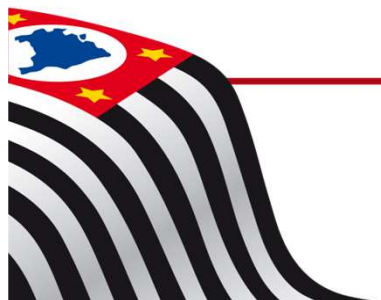


- 
1. Coloque em uma panela funda o leite condensado, a margarina e o chocolate em pó.
  2. Cozinhe [no fogão] em fogo médio e mexa sem parar com uma colher de pau.
  3. Cozinhe até que o brigadeiro comece a desgrudar da panela.
  4. Deixe esfriar bem, então unte as mãos com margarina, faça as bolinhas e envolva-as em chocolate granulado

Metodologia

Ferramenta

Processo

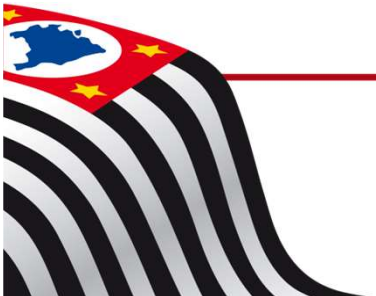




# Engenharia de Software: Elementos Fundamentais

- **Paradigma:** estilo de fazer algo, representa uma abordagem ou filosofia para a construção de software.

Ex.: cozinha francesa, chinesa, orientado a objetos, procedural.

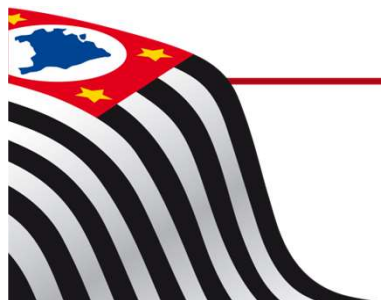
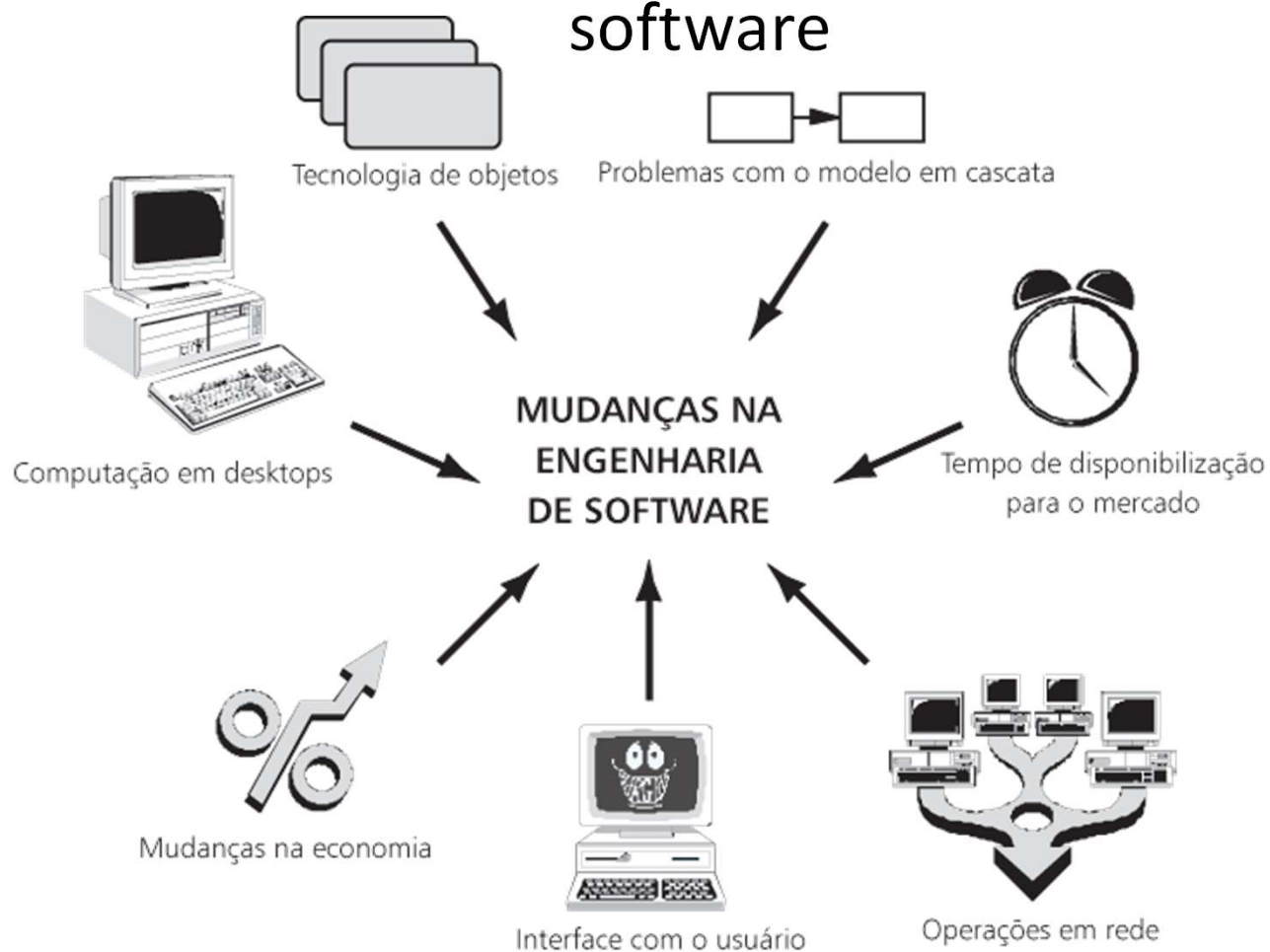


# Fatores-chave que mudaram a prática da engenharia de software

- Aspecto crítico do tempo para entrega do produto ao mercado, no caso de produtos comerciais
- Mudanças na economia da computação (redução dos custos de hardware e aumento nos custos de desenvolvimento e manutenção)
- Disponibilidade poderosa da computação em desktops
- Aumento das redes locais e remotas
- Disponibilidade e adoção da tecnologia orientada a objetos
- Uso de interfaces gráficas
- Imprevisibilidade do modelo de desenvolvimento de software cascata



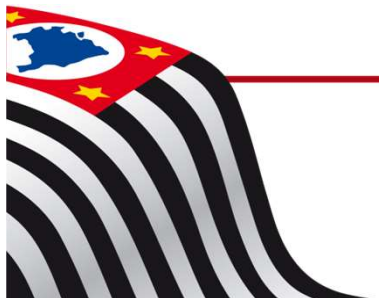
# Fatores-chave que mudaram a prática da engenharia de software



# Disciplina de engenharia de software

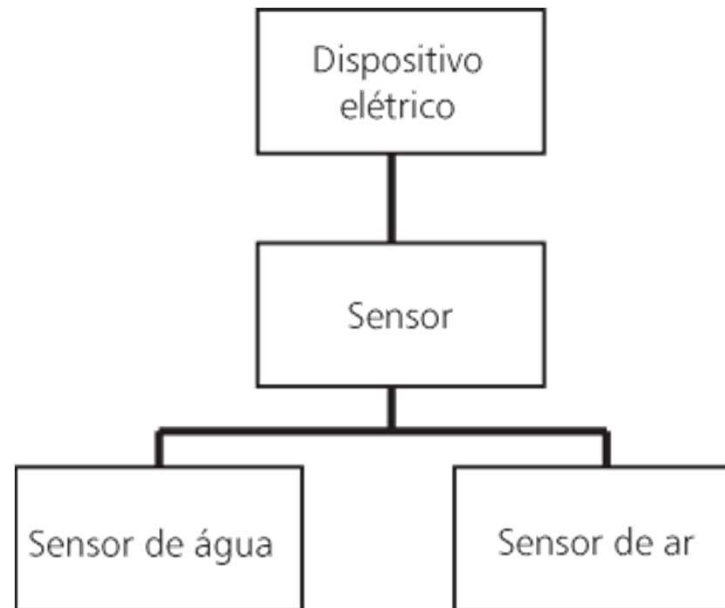
Oito noções fundamentais que formam a base de uma disciplina de engenharia de software efetiva:

- Abstração
- Métodos e notações de análise e projeto
- Protótipo da interface com o usuário
- Arquitetura de software
- Processo de software
- Reuso
- Medição
- Ferramentas e ambientes integrados

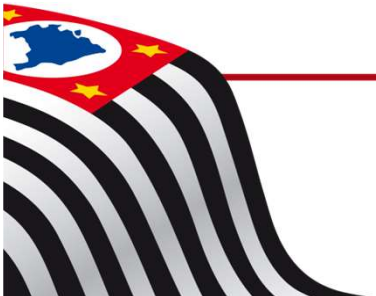


# Abstração

- Descrição de um problema com um nível de generalização que permite concentrar nos aspectos principais do problema, sem se perder nos detalhes

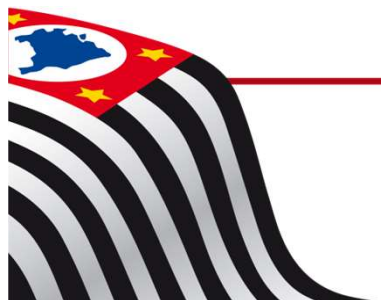


Hierarquia simples de monitoração de equipamento



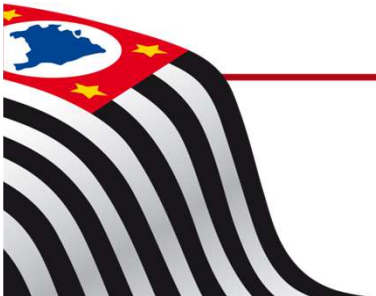
# Arquitetura de software

- A arquitetura de sistemas descreve o sistema em termos de um conjunto de unidades arquitetônicas e um mapa de como essas unidades se relacionam entre si.
- Wasserman (1996) mostra cinco modos de dividir o sistema em unidades:
  1. decomposição modular – baseada na atribuição de funções aos módulos
  2. decomposição orientada a dados – baseada em estruturas de dados externas
  3. decomposição orientada a eventos – baseada nos eventos com os quais o sistema deve lidar
  4. projeto ‘de fora para dentro’ – baseado nas entradas dos usuários no sistema
  5. projeto orientado a objetos – baseado na identificação de classes de objetos e suas inter-relações

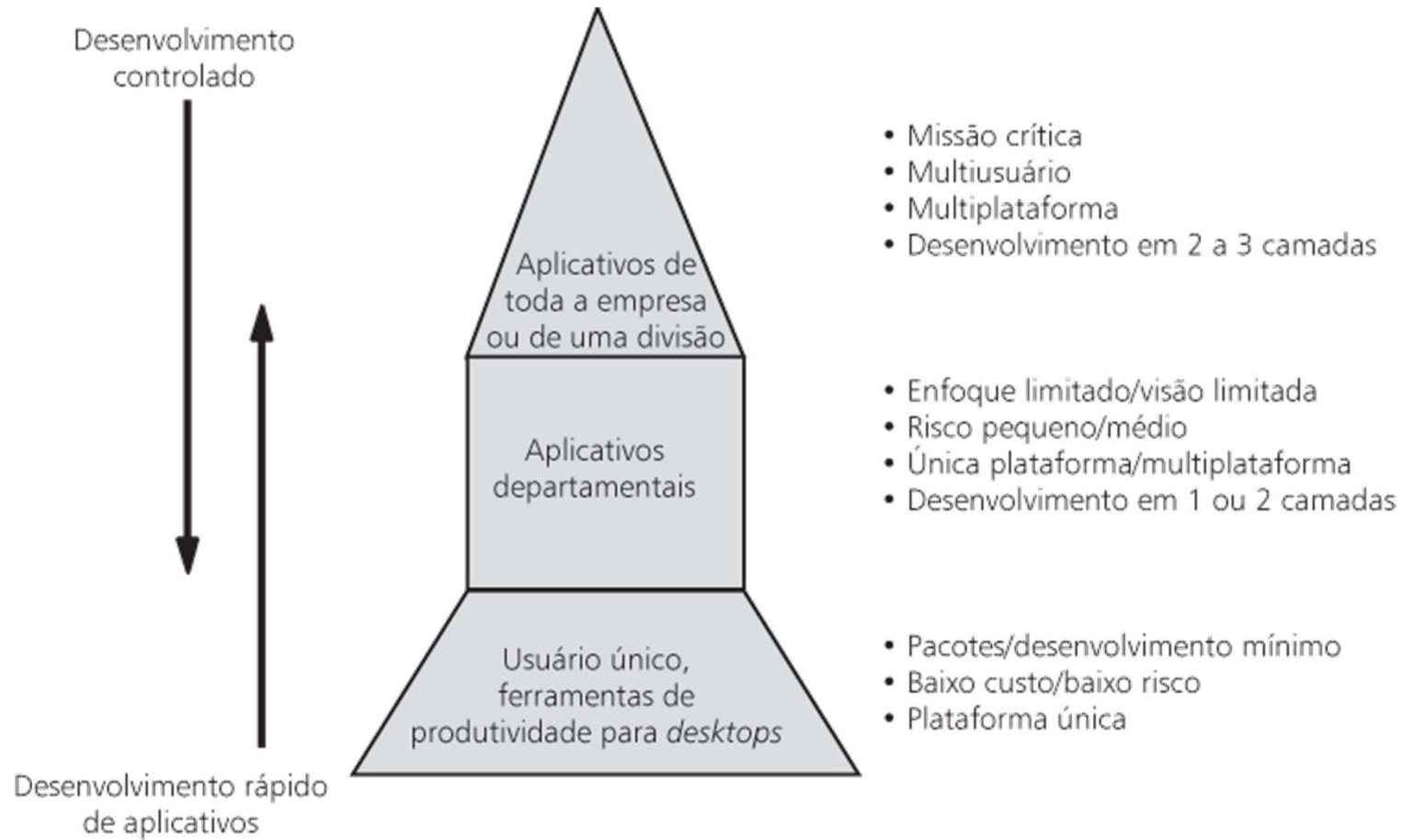


# Processo de desenvolvimento de software

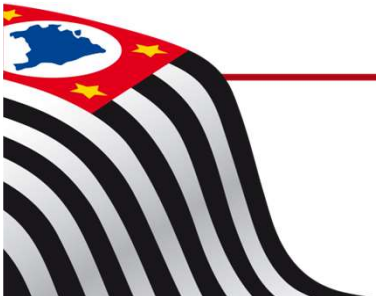
- Processo de desenvolvimento de software - qualquer descrição do desenvolvimento de software que contenha algumas das nove atividades ao lado, organizadas de tal modo que, juntas, produzam um código testado.
  - Análise e definição dos requisitos
  - Projeto do sistema
  - Projeto do programa
  - Escrever os programas
  - Testes das unidades
  - Teste de integração
  - Teste do sistema
  - Entrega do sistema
  - Manutenção



# Processo de desenvolvimento de software



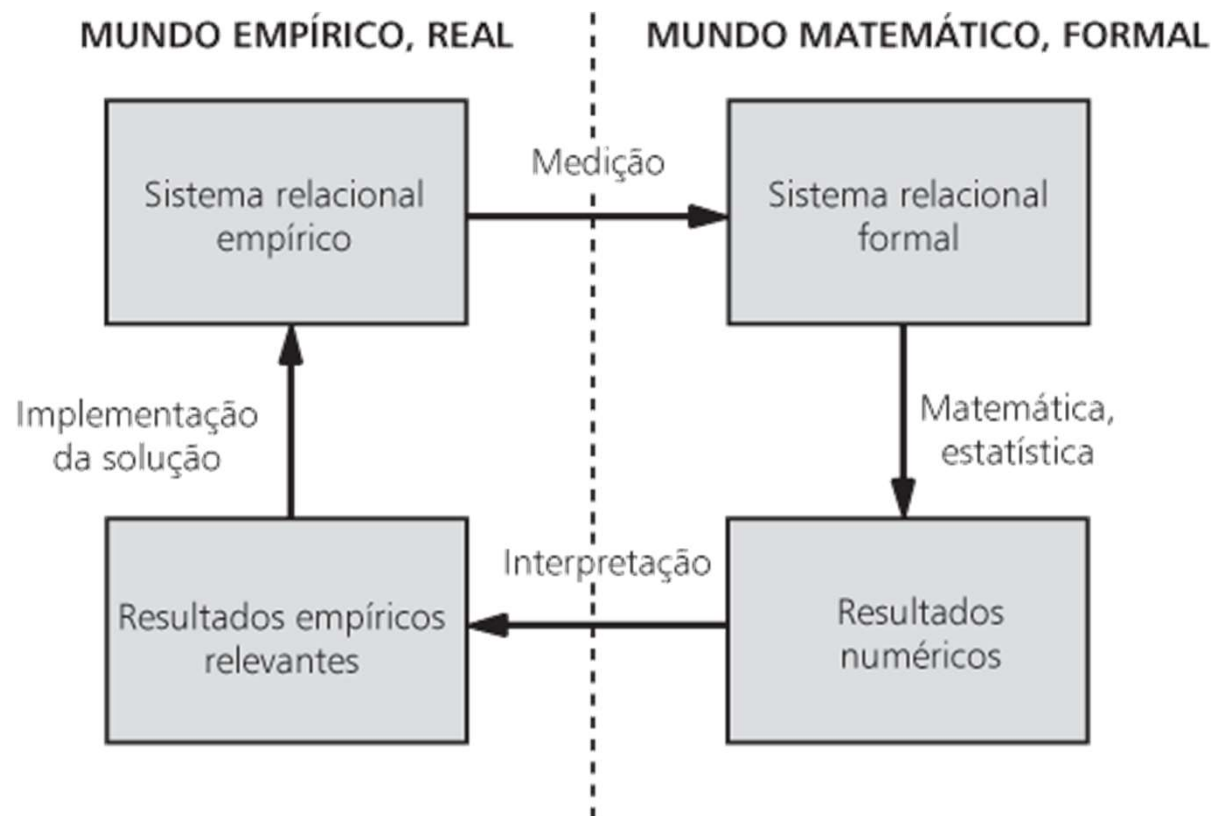
Diferenças no desenvolvimento (Wasserman, 1996)





# Medição

- Palavra-chave: aprimoramento



Utilizando medição para ajudar a encontrar uma solução

