



# ENGENHARIA DE SOFTWARE I

*Profº Me. Jones Artur Gonçalves*

# O QUE É UM SOFTWARE?

“Instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados” .Pressman, 1995

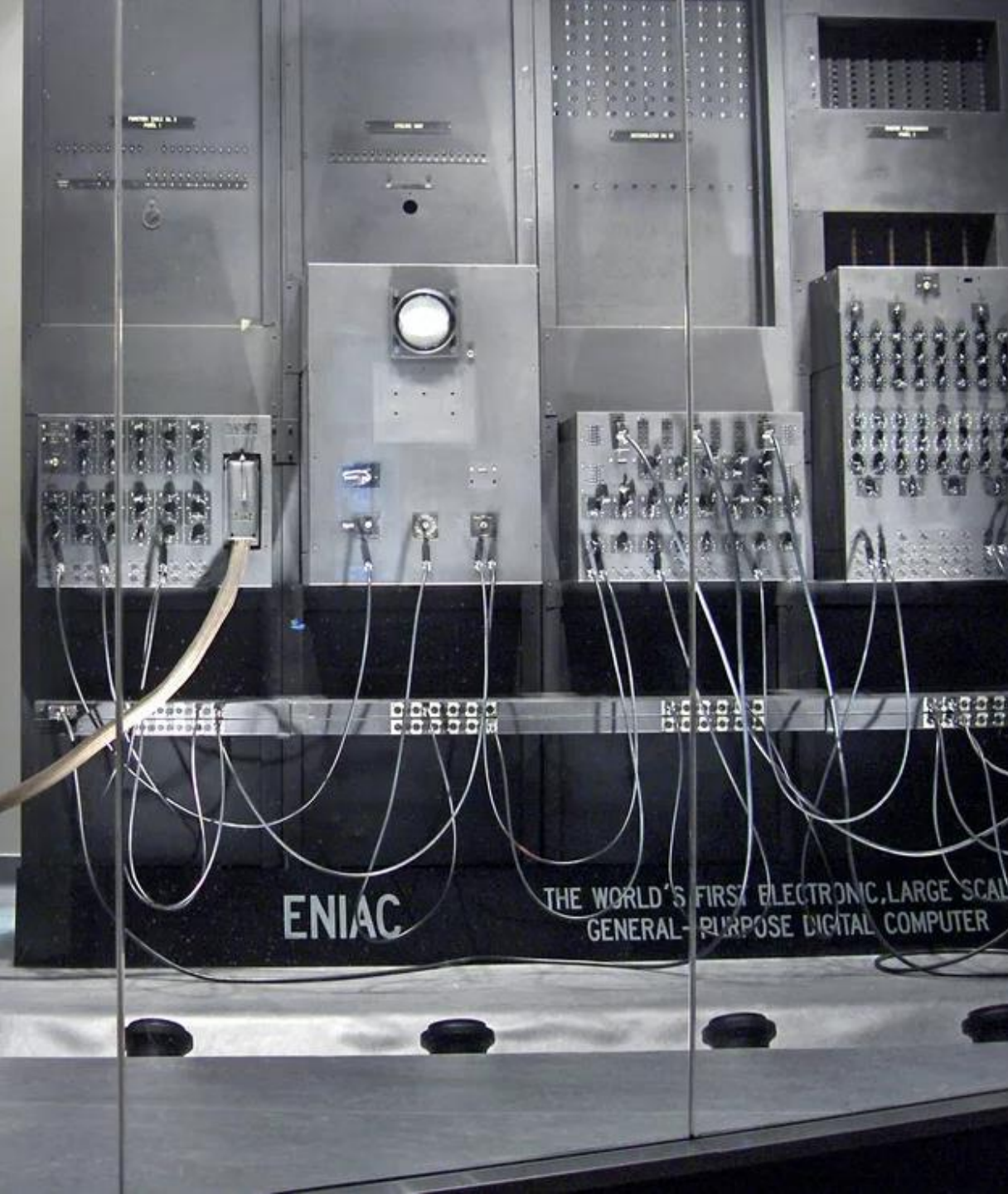
```
mirror_mod = modifier_ob.  
set mirror object to mirror  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
-- OPERATOR CLASSES --  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
context):  
context.active_object is not
```



# SOFTWARE

*Mais do que programas-fonte e executáveis:  
compreende também a documentação  
associada à ele, dados e configuração*





# 1950

## Os primeiros anos:

- 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

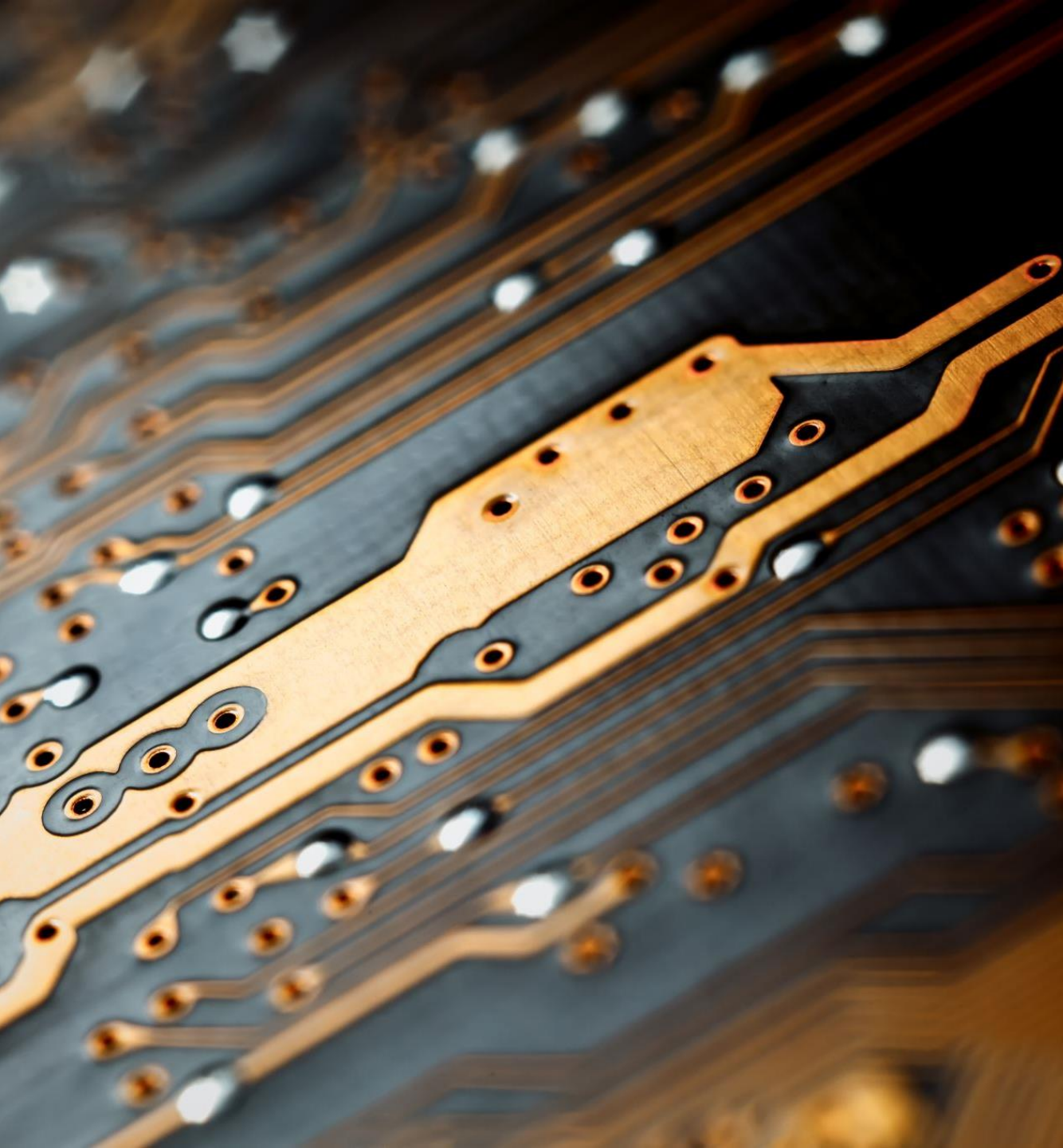




# 1960

**A segunda era:**

- Sistemas Multiusuários
- Tempo real
- Banco de dados
- Produtos de software



# 1970

## **A terceira era:**

- Sistemas Distribuídos
- Inteligência Embutida (microprocessadores)
- Hardware de baixo custo
- Impacto de Consumo



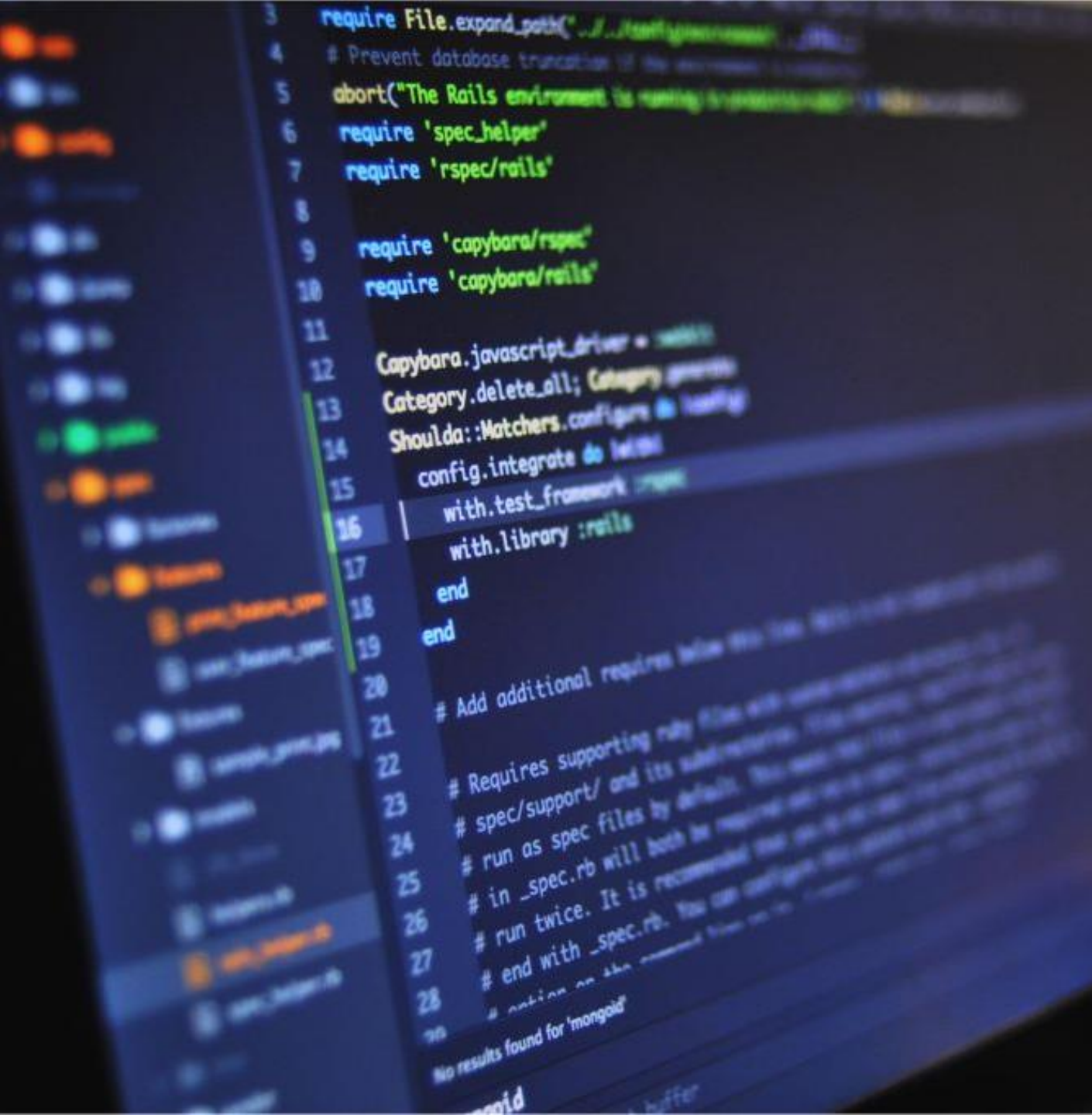


# 1980

## Quarta Era:

- Sistemas de Desktop poderosos
- Sistemas Especialistas
- Computação Paralela





**“O Software ultrapassou o Hardware como chave para o sucesso de muitos sistemas baseados em computador”  
(Pressman)**

# CAUSAS DA CRISE DO SOFTWARE

## (SURTIU NOS ANOS 70)

### O Início Da Engenharia De Software

Apesar do termo Engenharia de software já ter sido utilizado desde a década de 1950, ele só ficou conhecido depois de duas conferências patrocinadas pelo Comitê Científico da OTAN em 1968 e 1969.

A crise de software foi uma decorrência da imaturidade do mercado e dos profissionais de computação da época, pois a tecnologia vinha avançando e junto com ela o desenvolvimento de software cada vez mais complexos.

Com o rápido crescimento do poder computacional foi possível utilizar os computadores em tarefas cada vez mais complicadas, fazendo com que o que deveria ser um avanço para a sociedade, com o tempo foram se revelando uma série de problemas como o excesso de custos, dano a propriedade ocasionadas por falha de softwares.



# CAUSAS DA CRISE DO SOFTWARE

## **Crise: motivos**

- Falta de envolvimento do usuário;
- Análise e projeto inadequados;
- Falta de flexibilidade no projeto;
- Prazos longos;
- Elevada rotatividade de pessoal;

# CAUSAS DA CRISE DO SOFTWARE

## **Crise: motivos**

- Velocidade da mudança tecnológica;
- Dificuldade de formalização;
- Velocidade na mudança dos mercados;
- Velocidade na obsolescência dos sistemas;



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Denver International Airport



O 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

# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Denver International Airport

Erros no sistema automático de transporte de bagagens:

### Sistema deveria suportar:

- 21 milhas de trilhas;
- 4000 rotas de carros;
- 5000 olhos eletrônicos;
- 400 receptores de rádio - intercomunicação;
- 100 computadores conectados entre si;

### As consequências:

- Atraso na abertura do aeroporto com custo total estimado em US\$ 360 milhões
- 86 milhões para consertar o sistema





# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Foguete Ariane V

Projeto da Agência Espacial Europeia que custou:

- 10 anos.
- US\$ 8 Bilhões.
- Capacidade: 6 toneladas.

Voo inaugural em 04/06/96



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Foguete Ariane V

Resultado:

Explosão 40 segundos após a decolagem.

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



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Foguete Ariane V

Fato: o veículo detonou suas cargas explosivas de autodestruição e explodiu no ar. Por quê?

Porque ele estava se quebrando devido às forças aerodinâmicas. Mas por quê?

O foguete tinha perdido o controle de direção (altitude). Causa disso?

Os computadores principal e backup deram shutdown ao mesmo tempo



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: Foguete Ariane V

Por que o Shutdown? Ocorreria um run time error (out of range, overflow , ou outro) e ambos computadores se desligaram. De onde veio este erro?

**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.**

Mas, por quê???

O resultado desta conversão não era mais necessário após a decolagem...



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: O caso da FAA: Federal Aviation Administration - USA

O projeto:

- um novo sistema de controle de tráfego aéreo chamado AAS;
- programa: milhões de linhas de código distribuídas entre centenas de computadores com hardwares funcionando em tempo real;
- Contratada: IBM.

# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: O caso da FAA: Federal Aviation Administration - USA

### **Custo Estimado:**

- US\$ 500 por linha de código (5x a média de mercado).

### **Resultados:**

- FAA pagando entre \$700 e \$900 por linha de código;
- FAA cancelou duas das quatro partes do projeto
- atraso na entrega
- o sistema (cheio de Bugs) está em análise pela Carnegie Mellon e MIT



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: O caso Therac-25

Outro exemplo conhecidos quando referente a falha de software, é o caso Therac-25, que foi uma máquina de radioterapia controlada por computador que desenvolvida pela Atomic Energy of Canada Limited (AECL) que foi fabricada no ano de 1982, com o propósito de aplicar doses de radiação nos pacientes para tratamentos de câncer.



# CAUSAS DA CRISE DO SOFTWARE

## Crise do Software: O caso Therac-25

Esta máquina foi responsável por pelo menos 6 acidentes entre 1985 e 1987, quando pacientes receberam uma overdose de radiação, resultando em mortes ou ferimentos graves.

Pesquisadores que investigaram os acidentes encontraram diversas causas que contribuíra, para os acidentes acontecerem.

Entre elas, estavam vários erros no desenvolvimento do software, pois o software não havia sido testado junto do hardware e muito menos foi revisado ou documentando para que fosse entendido os motivos dos erros posteriormente.





Como o cliente explicou...



Como o líder de projeto entendeu...



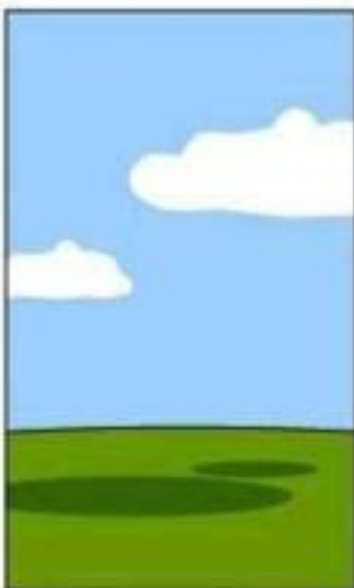
Como o analista projetou...



Como o programador construiu...



Como o Consultor de Negócios descreveu...



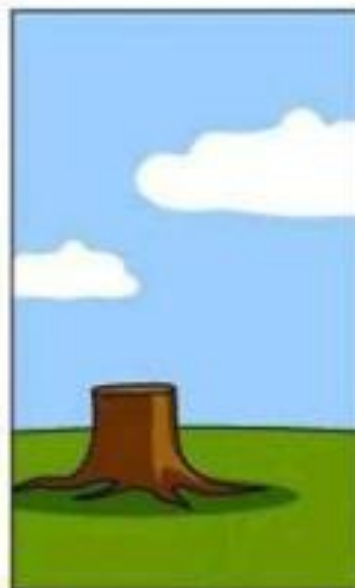
Como o projeto foi documentado...



Que funcionalidades foram instaladas...



Como o cliente foi cobrado...



Como foi mantido...



O que o cliente realmente queria...



# CRISE DO SOFTWARE

A criação da Engenharia de Software surgiu numa tentativa de contornar a crise do software e dar um tratamento de engenharia (mais sistemático e controlado) ao desenvolvimento de sistemas de software complexos.

The diagram illustrates a neural network architecture with four distinct layers:

- Input Layer:** The first layer on the left, containing 5 nodes.
- Multiple hidden Layers:** Two intermediate layers, each containing 5 nodes.
- Output Layer:** The final layer on the right, containing 5 nodes.

Connections are shown as lines between nodes in adjacent layers, representing the weights and biases of the network. The background features a dark blue grid with faint, glowing numbers and text, suggesting a digital or data environment.

# Engenharia de Software

preocupa-se com as teorias, os métodos e as ferramentas para o desenvolvimento profissional de software.

# Conceito

“... é a ciência e a arte de especificar, projetar, implementar e manter atualizados e corretos, com economia, em tempo útil e de forma elegante, programas, documentação e procedimentos operacionais para sistemas computacionais de utilidade para a humanidade.”

**A. Brown, A. Earl e J. McDermid**



# Conceito

“... é o estabelecimento e o uso de um conjunto de princípios de engenharia com o objetivo de construir software confiável, eficiente e economicamente viável em máquinas reais.”

**Bauer**

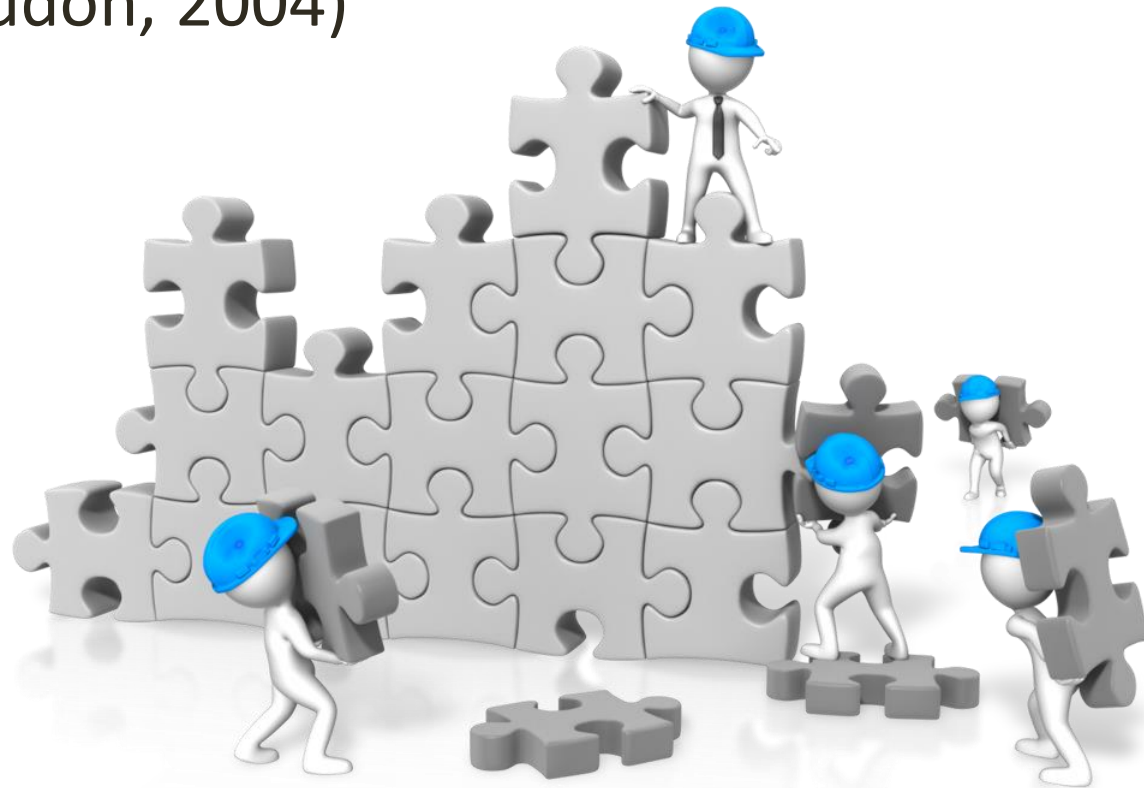
# O que é um sistema?



# O que é um sistema?

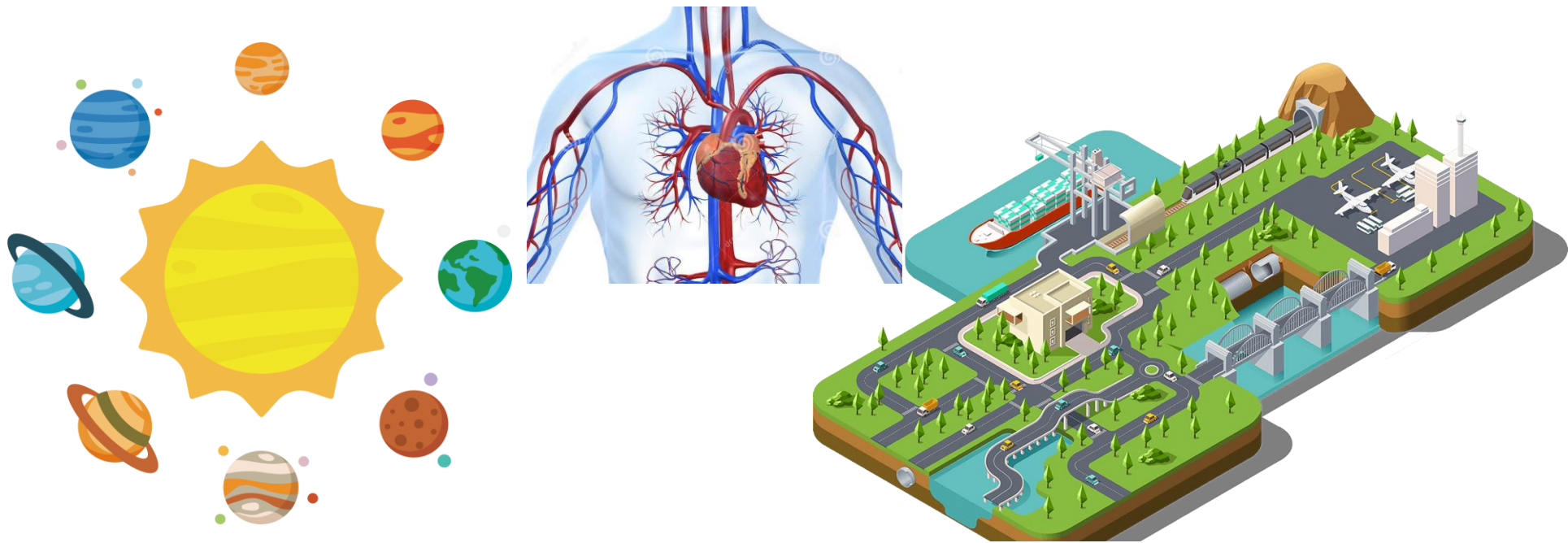
“Um conjunto de partes, componentes, que interagem entre si, de forma ordenada, a fim de atingir um objetivo comum”

(Stair, 1998;Laudon;Laudon, 2004)



# O que é um sistema?

O que há em comum entre o Sistema Solar, o Sistema Circulatório Humano e o Sistema de Transporte de uma cidade?





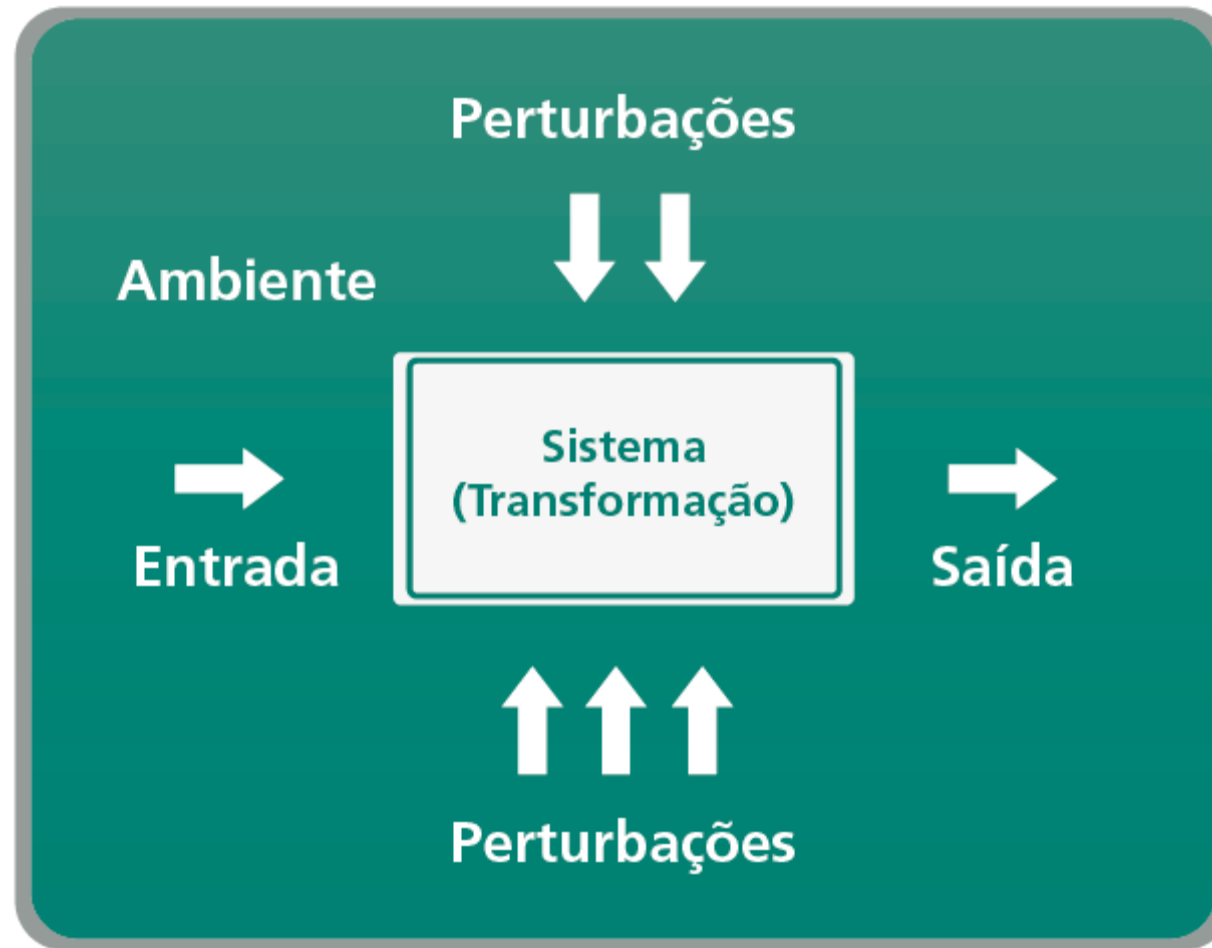
# O que é um sistema?

- Um dos conceitos que possui grande aceitação e aplicação para a nossa área de Computação é definido por Peter Schoderbek (SCHODERBEK et al., 1990 apud MARTINELLI; VENTURA, 2006, p. 6), segundo o qual:
- É o conjunto de **objetos**, com relações entre os objetos e os atributos relacionados com cada um deles e com o ambiente, de maneira a formar um todo”. (MARTINELLI; VENTURA, 2006, p. 6).

# O que é um sistema?

- **a) os objetos** são os elementos que compõem o **sistema**; os relacionamentos são as fronteiras que ligam os objetos;
- **b) os atributos** são as características tanto dos objetos como dos relacionamentos; o ambiente é o que está fora do sistema, ou seja, não participa do sistema, porém, ele está inserido nesse espaço delimitado ou não.

# O que é um sistema?



O conceito de sistemas graficamente apresentado.

# Considerações básicas sobre sistemas

A abordagem sistêmica, nos leva a considerar que é importante conhecermos algumas características dos sistemas.

Essas características nos ajudam a compreender melhor o tipo de sistema que estamos estudando.

No caso da Análise e Projeto de Sistemas, precisamos conhecer bem cada sistema no qual trabalhamos para que fique fácil criarmos soluções computacionais para eles.



# Considerações básicas sobre sistemas

Muitas vezes os problemas informados por nossos usuários são melhor compreendidos pelos programadores quando conseguimos identificar essas características:

- **a) o objetivo central do sistema e as respectivas medidas de rendimento;**
- **b) o ambiente do sistema;**
- **c) os recursos do sistema;**
- **d) a administração do sistema.**

# Considerações básicas sobre sistemas

Os **objetivos** significam aquilo que deve ser alcançado pelo sistema, ou seja, o motivo pelo qual um sistema existe e as **medidas de rendimento** indicam o quanto ele está (ou não) alcançando seus objetivos.

# Considerações básicas sobre sistemas

Esse rendimento pode ser medido de várias maneiras, por exemplo, em um sistema de controle de estoque que tenha como objetivo principal não deixar faltar produtos em um almoxarifado, a quantidade de produtos que não ficaram faltando nas prateleiras pode ser considerada uma medida de rendimento.

Em outras palavras, se faltar produtos nas prateleiras por erro nos relatórios do sistema, pode-se concluir que o programa não atendeu a seu objetivo, que era garantir sempre a disponibilidade de produtos.

# Considerações básicas sobre sistemas

O **ambiente do sistema** está associado com aquilo que está ao redor dele, ou seja, com todos os outros sistemas ou “coisas” que se encontram externamente próximos dele.

Os objetivos precisam de **recursos** que sejam executados.

No caso da Informática, os principais recursos que garantem o alcance dos objetivos são aqueles associados com a Computação, por exemplo: **recursos de *hardware*** (processador, memória, *clock*, etc.), **recursos de *software*** (sistema operacional, banco de dados, compiladores, etc.), **recursos humanos** (analistas, programadores, técnicos, etc.), entre outros que podem ser associados aos sistemas de informação.



# Considerações básicas sobre sistemas

A característica **administração do sistema** preocupa-se com as funções de planejamento e de controle associadas ao sistema.

Sem um correto planejamento que contemple todas as etapas de desenvolvimento de um sistema não é possível um trabalho de programação de soluções que atendam os nossos usuários.

É importante, também, que os administradores acompanhem o trabalho com os sistemas dando retorno aos interessados sobre todos os passos executados e aqueles que ainda não foram completados para a finalização do trabalho.

# O desenvolvimento de software

Os principais motivos que levam as organizações a desenvolverem sistemas são:

- o **aumento da qualidade dos produtos e serviços**, através da automatização das rotinas;
- a **redução de custos**;
- o **aumento da vantagem competitiva sobre os concorrentes**;
- o **aumento do nível de serviço, do desempenho dos recursos humanos**;
- a **melhoria do processo de tomada de decisões pela administração da empresa**.

# O desenvolvimento de software

Estudos feitos nos EUA para se medir a produtividade no desenvolvimento de sistemas (STANDISH GROUP, 2011) revelaram que:

- a) **30% a 40%** dos projetos são **cancelados**;
- b) **50%** dos projetos custam mais que o **dobro do custo** inicialmente projetado;
- c) **15 a 20%** dos projetos terminam **dentro do prazo e dentro do orçamento**.

Os principais problemas encontrados foram:

- a) **falta de qualidade do produto final**;
- b) **não cumprimento dos prazos**;
- c) **não cumprimento dos custos**.

# O desenvolvimento de software

A partir desses resultados surgiram estudos de **processos** e técnicas para melhoria da qualidade e confiabilidade dos produtos, e também a produtividade dos desenvolvedores.

Dessa forma a melhor maneira de se aumentar a qualidade do *software* é definindo um **processo** eficaz **de desenvolvimento** que contemple todas as fases necessárias para sua produção.

Outros requisitos importantes devem ser atendidos:

- **a) flexibilidade:** os sistemas devem ser facilmente alteráveis para garantir a sua evolução;
- **b) confiabilidade:** o número de defeitos deve ser o menor possível;
- **c) desempenho:** o sistema deve ter um desempenho razoável;
- **d) facilidade de uso:** o sistema deve ser de fácil utilização.



# CATEGORIAS DE SOFTWARE



## **Sistema Embarcado**

Embutido - eletrônicos -  
IoT



## **Software para negócios**

ERP



## **CAD**

3D para engenharias  
e arquitetura



## **Celular**

Tempo real+  
Geoprocessado+Embarca  
do+ Inteligência Artificial

# SOFTWARE = PRODUTO



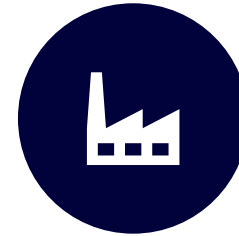
PROJETO (PRAZO,  
CUSTO E ESCOPO)



PROCESSO =  
METODOLOGIA (LOGIA  
= ESTUDO)



ESCOPO = DEFINIÇÕES  
DO SISTEMA



MÉTODOS = TÉCNICAS  
DE PRODUÇÃO (COMO  
FAZER)



FERRAMENTAS - COM O  
QUE FAZER

**Escopo** é a finalidade, o alvo, ou o intento que foi estabelecido como meta final. O **escopo** é o objetivo que se pretende atingir, é sinônimo de fim, propósito ou desígnio.

# PRODUTO E PROCESSO

Ambos são aspectos da Engenharia de Software


**Ênfase:** do produto para o processo;

**Necessidade:** entregar produtos de software de qualidade para os clientes, desenvolvidos por processos consistentes, bem gerenciados e com controle efetivo de custos.

# PRODUTO E PROCESSO

# Processo

## Conjunto de atividades:

- bem definidas;
  - com responsáveis;
  - com artefatos de entrada e saída;
  - com dependências entre as mesmas e ordem de execução;
  - com modelo de ciclo de vida.
- 
- An illustration on the right side of the slide. It features a large, stylized hand in a light tan color holding a round clock with a teal border and a white face. The clock has two simple black hands. To the right of the hand, there are several small, semi-transparent icons: a teal arrow pointing down, a grey arrow pointing up, and a small document icon with a red header and yellow body. The background of the illustration is a solid teal color.





# PROCESSO DE SOFTWARE

Um conjunto de atividades cujo objetivo é o desenvolvimento ou evolução do software;

- Conjunto coerente de atividades para especificação, projeto, implementação e teste de sistemas de software;
- Atividades que são envolvidas no desenvolvimento de produtos de software.



# PRODUTO E PROCESSO

## **Produto:**

- Qualidade
- Atendimento à expectativas do cliente

## **Processo:**

- Consistente
- Bem gerenciado
- Custo e tempo controlados

# MÉTODOS

Descrição sistemática de como deve-se realizar uma determinada atividade ou tarefa;

A descrição é normalmente feita através de padrões e guias;



# MÉTODOS

Fornecem a técnica de “como fazer” para construir softwares.

Abrangem um amplo conjunto de tarefas para cada uma das fases propostas por um processo adotado.

Por exemplo, dentro da fase de Modelagem do processo, diferentes tipos de modelos podem ser adotados (UML, DFD, Etc).



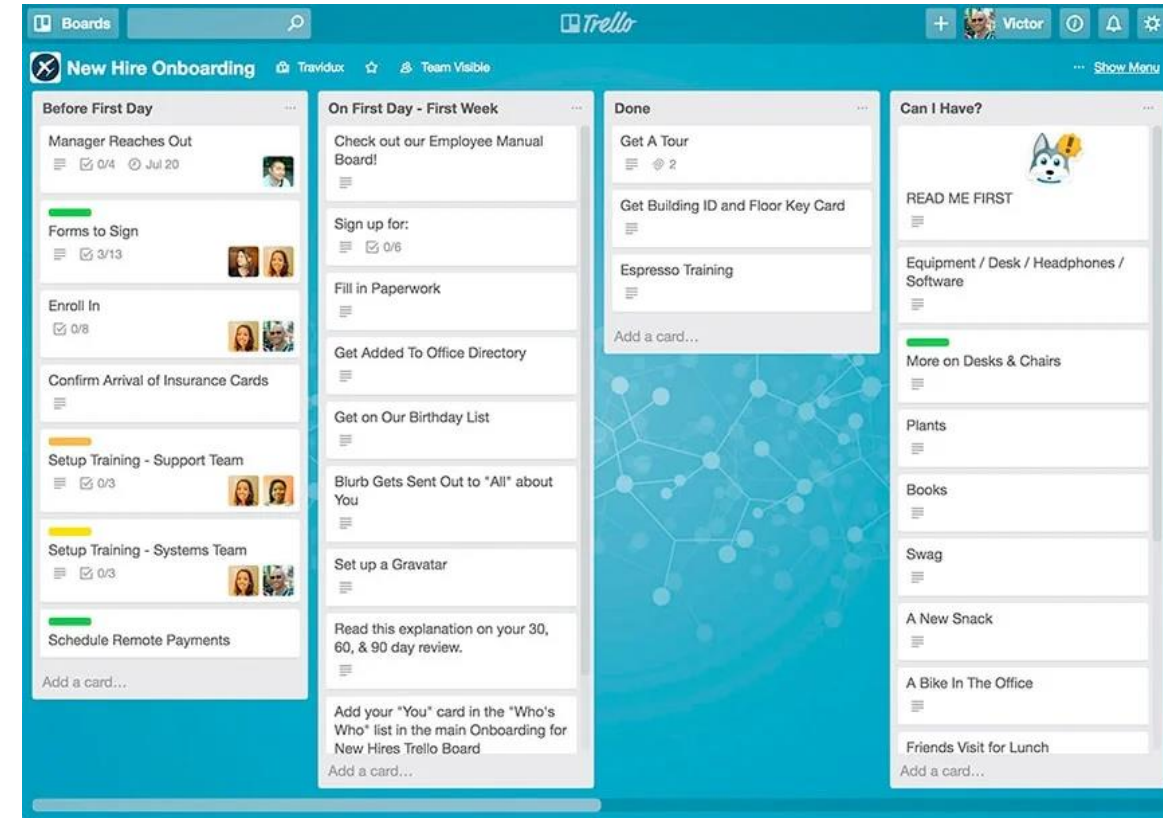


# FERRAMENTAS

Provê suporte computacional a um determinado método ou linguagem;

Ambiente de desenvolvimento: conjunto de ferramentas integradas (CASE: Computer Aided Software Engineering.);

Exemplos: Rational Rose, JBuilder, Enterprise Architect, Trello, Er-win.



# Software = Produto/Serviço



- Objetivo: **Qualidade**
- Engenharia em camadas:
  - Foco na Qualidade
  - Processo(Metodologias)
  - Métodos(Técnica)
  - Ferramentas(automatizadas ou não)

Segundo Pressman, a **Engenharia de Software** abrange um processo, um conjunto de métodos (práticas) e ferramentas que possibilitam aos profissionais desenvolverem **software** de altíssima qualidade

Figura 1: Camadas da Engenharia de Software.  
Fonte: PRESSMAN (2010).



O que estes eventos tem em comum?



# O que é um projeto ?



✓ Um Projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo.

*Guia PMBOK®*





# Desafios e Oportunidades

✓ US\$ 10 trilhões são gastos anualmente no mundo em projetos, o que equivale a aproximadamente 25% do PIB mundial



✓ Cerca de 16.5 milhões de profissionais estão envolvidos diretamente com gerenciamento de projetos no mundo.



PROJETO

Planejamento

Detalhado

# BIBLIOGRAFIA



## BÁSICA:

BEZERRA, EDUARDO. PRINCÍPIOS DE ANÁLISE E PROJETO DE SISTEMAS COM UML. 3 ED. RIO DE JANEIRO: ELSEVIER, 2015.

PRESSMAN, ROGER; MAXIM, BRUCE. ENGENHARIA DE SOFTWARE. 8 ED. SÃO PAULO: MCGRAW HILL BRASIL, 2016.

SOMMERVILLE, IAN. ENGENHARIA DE SOFTWARE. 10 ED. SÃO PAULO: PEARSON BRASIL, 2019.

## COMPLEMENTAR:

LARMAN, Craig. Utilizando UML e padrões. 3 ed. Porto Alegre: Bookman, 2007.

REZENDE, Denis Alcides. Engenharia de software e sistemas de informação. 3 ed. Rio de Janeiro: Brasport, 2005

# Referências

---

[HTTPS://PM2ALL.BLOGSPOT.COM/2015/05/A-CRISE-NO-DESENVOLVIMENTO-DE-SOFTWARE.HTML](https://pm2all.blogspot.com/2015/05/a-crise-no-desenvolvimento-de-software.html) DATA DE ACESSO: 01/07/2015.

[HTTP://WWW.ALEXANDRE.ELETRICA.UFU.BR/ESOF/AULA01.PDF](http://www.alexandre.eletrica.ufu.br/esof/aula01.pdf) DATA DE ACESSO: 01/07/2023

[HTTPS://WWW.DIO.ME/ARTICLES/DESCUBRA-O-QUE-E-A-CRISE-DO-SOFTWARE](https://www.dio.me/articles/Descubra-o-que-e-a-crise-do-software) DATA DE ACESSO: 05/12/2023

[HTTPS://CELSOKITAMURA.COM.BR/A-CRISE-DO-SOFTWARE-O-INICIO-DA-ENGENHARIA-DE-SOFTWARE/](https://celsokitamura.com.br/a-crise-do-software-o-inicio-da-engenharia-de-software/) DATA DE ACESSO: 05/12/2023

[HTTPS://SILO.TIPS/DOWNLOAD/ENGENHARIA-DE-SOFTWARE-01-INTRODUAO-MARCIO-DANIEL-PUNTEL](https://siilo.tips/download/engenharia-de-software-01-introducao-marcio-daniel-punzel) DATA DE ACESSO: 05/12/2023

KOTONYA, GERALD; SOMMERVILLE, IAN; REQUERIMENTS ENGINEERING: PROCESSES AND TECHNIQUES . JOHN WILEY & SONS, 1998.

# Referências

---

[HTTPS://PM2ALL.BLOGSPOT.COM/2015/05/A-CRISE-NO-DESENVOLVIMENTO-DE-SOFTWARE.HTML](https://pm2all.blogspot.com/2015/05/a-crise-no-desenvolvimento-de-software.html) DATA DE ACESSO: 01/07/2015.

[HTTP://WWW.ALEXANDRE.ELETRICA.UFU.BR/ESOF/AULA01.PDF](http://www.alexandre.eletrica.ufu.br/esof/aula01.pdf) DATA DE ACESSO: 01/07/2023

[HTTP://WWW.ALEXANDRE.ELETRICA.UFU.BR/ESOF/AULA02.PDF](http://www.alexandre.eletrica.ufu.br/esof/aula02.pdf) DATA DE ACESSO: 01/07/2023

[HTTPS://WWW.DIO.ME/ARTICLES/DESCUBRA-O-QUE-E-A-CRISE-DO-SOFTWARE](https://www.dio.me/articles/Descubra-o-que-e-a-crise-do-software) DATA DE ACESSO: 05/12/2023

[HTTPS://CELSOKITAMURA.COM.BR/A-CRISE-DO-SOFTWARE-O-INICIO-DA-ENGENHARIA-DE-SOFTWARE/](https://celsokitamura.com.br/a-crise-do-software-o-inicio-da-engenharia-de-software/) DATA DE ACESSO: 05/12/2023

BASEADO NOS SLIDES DE GLAUCO TODESCO, TIAGO VINÍCIUS E GLEIBSON RODRIGO, JAIR C LEITE, ALEXANDRE VASCONCELOS, DANIEL VILLELA, MARIANA VIEIRA.

[HTTPS://SILO.TIPS/DOWNLOAD/ENGENHARIA-DE-SOFTWARE-01-INTRODUAO-MARCIO-DANIEL-PUNTEL](https://siilo.tips/download/engenharia-de-software-01-introducao-marcio-daniel-punzel) DATA DE ACESSO: 05/12/2023

KOTONYA, GERALD; SOMMERVILLE, IAN; REQUERIMENTS ENGINEERING: PROCESSES AND TECHNIQUES . JOHN WILEY & SONS, 1998.



