



# BandTec

DIGITAL SCHOOL



# Engenharia de Software

## Aula 1 – Introdução a Engenharia de Software

Professor Esp. Gerson Santos

# Objetivo da Aula

- Curso de ADS. Tirar dúvidas. (Coordenação)
- Introdução
- Entender o que é e para que serve Engenharia de Software
- Iniciar UX. Hoje já tem atividade.

## CORE

Coding e experimentação

## TECNOLOGIA

Métodos, práticas e produtos

## CONCEITO

PPC - Diretrizes –ENADE

## TRENDS

Demandas de mercado

### 1º SEMESTRE

Algoritmos

P&I / Socio

JavaScript, HTML, CSS,  
Bootstrap, Azure, SQL,  
Scrum, ITIL, Arduino,  
GitHub.

BD, TI e Arq Comp  
Processos de Negócio  
Lógica e Estatística  
*Socioambiental*

Introdução Node JS  
IoT

### 2º SEMESTRE

Ling. Prog.

P&I / Socio

Java, Maven, API, UML,  
BPMN, Canvas, Design  
Sprint, VMs, Windows,  
Linux, Container.

An. Sistemas e SO  
POO (Orient. Objetos)  
Gerência de Projetos  
Engenharia de Requisitos

Dashboards

### 3º SEMESTRE

Web

P&I / Socio

React, Spring MVC,  
JSON/REST, JUnit,  
Microserviços, UX,.

Eng. de Soft. e Estr. de Dados  
Ciclo de vida do SW  
Vetores, Pilhas, Recursão..  
Design de Interfaces

Spock (testes)  
Figma

### 4º SEMESTRE

App

P&I / Socio

Android, Kotlin, AWS,  
Computação em  
Nuvem (NFV, SDN e  
Serv. de Armaz)

Sistemas Distribuídos  
Ética e Empreendedorismo,  
Conceitos de Redes e SI

DevOps  
Cyber Security

DESCOBRIR

EXPERIMENTAR  
PLANEJAR

RESPONSABILIDADE  
PROFISSIONALISMO

AUTONOMIA  
INOVAÇÃO

# Disciplinas no 3º semestre – Turma A



## Zoom no BootCamp

- Cada tema dura 1 sprint
- Aulas práticas
- Avaliação é o trabalho realizado em aula
- Nota considerada em PI
- Aula normal!



# Disciplinas no 3º semestre – Turma B



## Zoom no BootCamp

- Cada tema dura 1 sprint
- Aulas práticas
- Avaliação é o trabalho realizado em aula
- Nota considerada em PI
- Aula normal!



# Objetivo da Aula – Agora como Professor

- Introdução a Disciplina
- Entender o que é e para que serve Engenharia de Software
- UX vs UI

# Expectativas com a disciplina

$$\text{SATISFAÇÃO} = \frac{\text{REALIDADE}}{\text{EXPECTATIVA}}$$



# Importante

- Estamos aqui para aprender;
- Respeitar as diferenças e individualidades;
- Somos um time;
- Temos a sorte de estar juntos aqui!
  - Sorte = Oportunidade + Competência

# Regras básicas

- Notebooks fechados enquanto o professor apresenta o conteúdo
- Celulares em modo silencioso e guardado, para não tirar sua atenção
- Se, caso haja uma situação urgente e você precisar atender ao celular, peça licença para sair da sala e atenda fora da aula.
- As aulas podem e devem ser divertidas mas :
  - Devemos respeitar uns aos outros – cuidado com as brincadeiras.
  - Foco total no aprendizado, pois nosso tempo em sala de aula é precioso.
  - Venham sempre com o conteúdo da aula passada em mente e as atividades realizadas.
  - Procure ir além daquilo que lhe foi proposto.
  - Capricho, apresentação e profundidade no assunto serão observados.



Importante :

- Realizar/Entregar todas as avaliações continuadas
- Realizar/Entregar os exercícios
- Temos 3h de Aula. O tempo é suficiente, mas não haverá tempo para revisões, então, evitem faltas.

- Não é uma disciplina focada em certificações
- Faremos o possível para deixar a disciplina mais prática;
- O combinado deve ser feito
- 0 – não sei ; 1 – sei ; null Vou saber/vou preparar (Não vamos trabalhar na enrolação)
- O tempo de aula expositiva precisa de FOCO & PARTICIPAÇÃO, sempre vai ter tempo para as outras coisas
- Se não estiver na sala....perde muito conteúdo...(isso vale para o espírito também)

## Bibliografia:

Engenharia de Software 8º Edição / Ian Sommerville

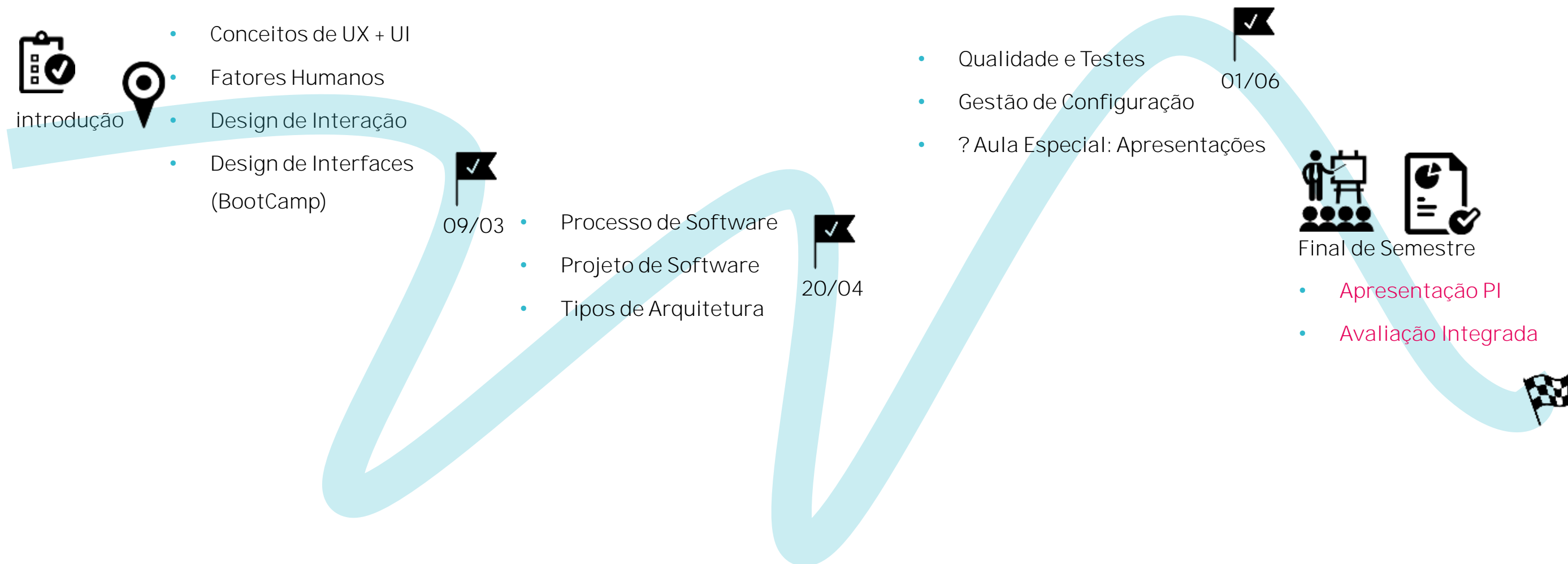
Engenharia de Software 6º Edição / Roger Pressman



## Adicional

- Code Complete
- SWEBOK
- Entre MUITOS outros que vou informar em tempo de aula.

# Engenharia de Software– Nosso caminho



## LEGENDA

- Conteúdo
- Entregável PI
- ✓ Conteúdo Finalizado
- ✓ Entregável Finalizado



Onde Estamos



Semana final das Sprints  
Semana das Entregas de PI

# O que é Engenharia de Software?

- ISO/IEC/IEEE Systems and Software Engineering  
“A aplicação sistemática, disciplinada e quantificáveis abordagens para o desenvolvimento, operação e manutenção de software”
- É uma disciplina de engenharia relacionada a todos os aspectos de produção de software (Sommerville)
- Engenharia de software engloba processos, métodos e ferramentas que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade. (Pressman)
- en·ge·nha·ri·a (engenho + -aria) substantivo feminino
  1. Conjunto de técnicas e métodos para aplicar o conhecimento técnico e científico na planificação, criação e manutenção de estruturas, máquinas e sistemas para benefício do ser humano.
  2. Ciência ou arte da construção (ex.: engenharia mecânica, engenharia militar, engenharia naval)

# A natureza do Software

Software é tanto o produto quanto o meio que distribui o produto.

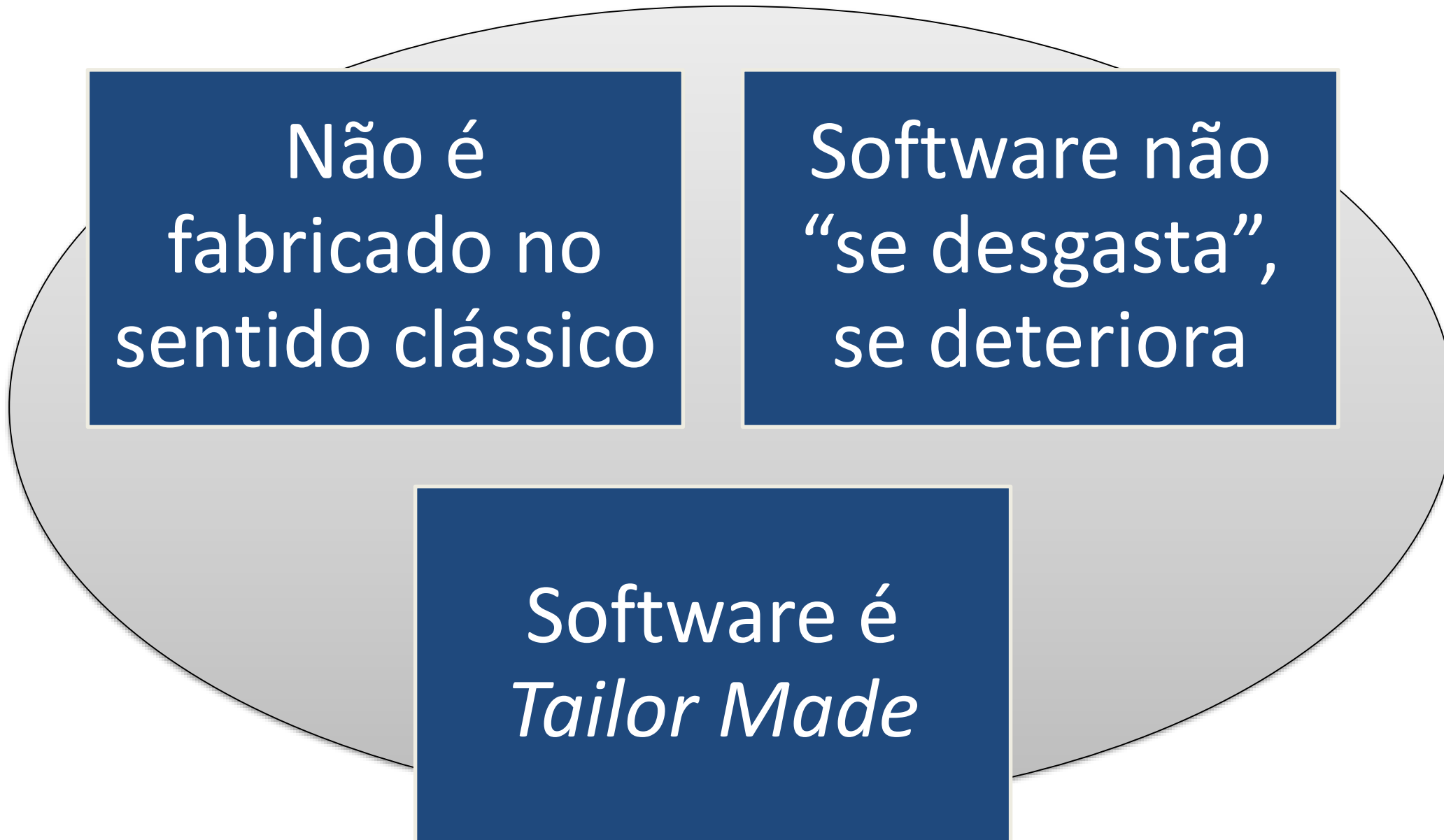
**“O software é um lugar onde os sonhos são plantados e pesadelos são colhidos, um pântano abstrato e místico onde demônios terríveis competem com mágicas panaceias, um mundo de lobisomens e balas de prata.” Brad J. Cox (Criador da Objective C language)**

O software distribui o produto mais importante destes tempos, a INFORMAÇÃO.





# 0 software...



# Nosso Objetivo

Aprender/Ensinar processos, métodos e ferramentas para construção e manutenção de softwares profissionais.

# Profissional vs Amador vs Artesanal

- Amador
  - 1. Que ou o que ama; que ou o que gosta muito de algo ou de alguém. = AMANTE, APRECIADOR
  - 2. Que ou aquele que, por gosto e não por profissão, exerce qualquer ofício ou arte.
  - 3. Que ou o que revela inexperiência em algum assunto ou atividade.
  - 4. Que é praticado ou exercido por gosto e não profissionalmente (ex.: desporto amador; teatro amador).
- Profissional
  - 1. Que se relaciona com uma dada profissão (ex.: sindicato profissional, ensino profissional).
  - 2. Pessoa que faz uma coisa por ofício (ex.: profissionais do futebol). ≠ **AMADOR**
  - 3. Que ou quem revela profissionalismo. ≠ **ANTIPROFISSIONAL**
- Artesanal
  - 1. Relativo a artesanato.
  - 2. Que é fabricado por artesão.
  - 3. Que é feito sem recurso a meios sofisticados ou a técnicas elaboradas ou industriais.

## 1. Problemas no Mariner (1962)

Custo: 18,5 milhões dólares

Desastre: Mariner, um foguete com uma sonda espacial para Vênus, foi desviado de seu percurso de voo logo após o lançamento. O controle da missão destruiu o foguete 293 segundos após a decolagem.

Causa: Um programador, ao passar para o computador uma fórmula que haviam lhe entregado escrita manualmente, se esqueceu de uma barra. Sem ela, o software tratava variações normais de velocidade como se fossem sérios problemas, causando falhas por tentativas de correções que acabaram por enviar o foguete fora do curso.

## 2. Hartford Coliseu Desmorona (1978)

Custo: 70 milhões de dólares, além de outros danos de 20 milhões para a economia local

Desastre: Poucas horas depois de milhares de fãs deixarem o Coliseu Hartford, o teto de treliça de aço desabou sob o peso da neve molhada.

Causa: O programador do software CAD, utilizado para projetar o coliseu, incorretamente assumiu que o suporte do telhado de aço enfrentaria apenas compressão natural. Mas quando um dos suportes inesperadamente recebeu um bloco de neve, este desencadeou uma reação em cadeia que derrubou o telhado de outras seções como dominós.

## 3. CIA distribui gás aos soviéticos (1982)

Custo: Milhões de dólares e danos significativos a economia soviética

Desastre: O software de controle se descontrolou e produziu uma intensa pressão no gasoduto Trans-Siberian, resultando na maior explosão não-nuclear da história.

Causa: CIA operatives allegedly planted a bug in a Canadian computer system purchased by the Soviets to control their gas pipelines. The purchase was part of a strategic Soviet plan to steal or covertly obtain sensitive U.S. technology. When the CIA discovered the purchase, they sabotaged the software so that it would pass Soviet inspection but fail in operation.

## 4. 3ª Guerra Mundial (Quase!) (1983)

Custo: Quase toda a humanidade

Desastre: O sistema de alerta precoce soviético falsamente indicou que os Estados Unidos tinham lançado cinco mísseis balísticos. Felizmente, o oficial de serviço soviético tinha uma "**sensação** esquisita no **estômago**" e fundamentalmente, se os EUA estavam realmente atacando, eles lançariam mais de cinco mísseis, por isso ele relatou o aparente ataque como um alarme falso.

Causa: Um bug no software soviético falhou ao detectar reflexos solares como falsos mísseis.

## 5. Máquina medicinal mata (1985)

Custo: Três mortos e três seriamente feridos

Desastre: A máquina de radiação canadense Therac-25 irradiou doses letais em pacientes.

Causa: Por causa de um bug sutil chamado de "*condição de corrida*", um técnico acidentalmente configurou o Therac-25 de modo que o feixe de elétrons seria como um fogo de alta potência.

## 6. Crash na Wall Street (1987)

Custo: \$500 bilhões em um dia

Desastre: Em 19 de outubro de 1987, o índice Dow Jones caiu 508 pontos, perdendo 22,6% de seu valor total. Esta foi a maior perda que Wall Street já sofreu em um único dia.

Causa: Um mercado em grande alta foi interrompido por uma série de investigações conduzidas pela SEC e por outras forças do mercado. Como os investidores fugiram de ações investigadas, um número muito grande de ordens de venda foram gerados pelos computadores, quebrando sistemas e deixando os investidores efetivamente cegos.

## 7. Linhas da AT&T "**morrem**" (1990)

Custo: 75 milhões de ligações perdidas e 200 reservas aéreas perdidas

Desastre: Um switch dos 114 centros de switches da AT&T sofreu um problema mecânico que fez com que todo o seu centro fosse desligado. Quando o seu centro voltou a ativa, enviou uma mensagem aos outros, o que causou o desligamento dos outros centros e deixou a empresa parada por 9 horas.

Causa: Uma única linha de código em uma atualização de software implementada para acelerar chamadas causou um efeito cascata que desligou a rede.

## 8. Patriot Acaba com Soldados (1991)

Custo: 28 soldados mortos e 100 feridos.

Desastre: Durante a primeira Guerra do Golfo, um sistema (Patriot) americano de mísseis na Arábia Saudita falhou ao interceptar um míssil vindo do Iraque. O míssil destruiu acampamentos americanos.

Causa: Um erro de arredondamento no software calculou incorretamente o tempo, fazendo com que o sistema Patriot ignorasse os mísseis Scud de entrada.

## 9. Pentium Falha em uma Divisão Longa (1993)

Custo: \$475 milhões e a credibilidade de uma empresa

Desastre: O altamente promovido Pentium, da Intel, ocasionalmente cometeu erros ao dividir números de ponto flutuante em um intervalo específico. Por exemplo, dividindo  $4195835.0/3145727.0$  obteve 1,33374 ao invés de 1,33382, um erro de 0,006%. Embora o bug afetasse apenas alguns usuários, se tornou um pesadelo nas relações públicas. Com uma estimativa de 5 milhões de chips defeituosos em circulação, a Intel se ofereceu para substituir os chips Pentium apenas para os consumidores que poderiam provar que eles precisavam de alta precisão. Contudo a Intel acabou substituindo os chips de qualquer um que reclamou.

Causa: O divisor na unidade de ponto flutuante do Pentium tinha uma tabela de divisão falha, faltando cerca de cinco mil entradas, resultando nestes erros de arredondamento.

## 10. Ariane Rocket Goes Boom (1996)

Custo: \$500 milhões

Desastre: Ariane 5, o mais novo foguete da Europa não-tripulado, fora intencionalmente destruído segundos após seu lançamento em seu voo inaugural. Também foram destruídos quatro satélites científicos para estudar como o campo magnético da Terra interage com os ventos solares.

Causa: O desligamento ocorreu quando o computador de orientação tentou converter a velocidade do foguete de 64-bits para um formato de 16 bits. O número era muito grande, o que resultou em erro de estouro. Quando o sistema de orientação desligou, o controle passou para uma unidade idêntica redundante, que também falhou porque nele estava correndo o mesmo algoritmo.

## 11. Bug do Milênio (1999)

Custo: \$500 bilhões

Desastre: O desastre de um homem é a fortuna de outro, como demonstra o Bug do Milênio. Empresas gastaram bilhões com programadores para corrigir uma falha no software legado. Embora nenhuma falha significativa tenha ocorrida, a preparação para o Bug do Milênio teve um custo significativo e impacto no tempo em todas as indústrias que usam a tecnologia computacional.

Causa: Para economizar espaço de armazenamento de computador, softwares legados muitas vezes armazenavam anos para datas com números de dois dígitos, como 99 para 1999. Esses softwares também interpretavam 00 para significar 1900, em vez de 2000, por isso, quando o ano de 2000 veio, bugs apareceriam.

# Sintomas de imaturidade de Software

1. As pessoas não recebem o treinamento necessário;
2. Os projetos não são claros. (Não sabe se é manutenção, quem é responsável...);
3. As ferramentas não são suficientes. (cada um usa o que quer, não há ferramenta...);
4. Os procedimentos e padrões não existem ou existem só no papel (burocrático);
5. Os compromissos não são realistas (política, falta de competência, consequência da falta de processos...);
6. Forças caóticas. (Magias, Balas de prata, Gurus (infalíveis), Heróis (benévolos);
7. Problemas de escala. (Falta de documentação, ausência de desenho, complexidade desnecessária);
8. Problemas de comunicação. (Exemplo das bolinhas...mais para frente);



# Perguntas e Respostas - Glossário

O que é Software?	Programas de computador e documentação associada.
Engenharia de Software vs Engenharia de Sistemas	Engenharia de Sistemas engloba todos os aspectos, incluindo hardware e engenharia de processos. Eng. de sistema contém engenharia de software.
O que é um processo de Software?	Um conjunto de atividades cujo objetivo é o desenvolvimento ou evolução de um software. (Especificação, desenvolvimento, validação e evolução).
O que é um modelo de processo de Software?	Representação simplificada do processo de software. Ex: WorkFlow, Fluxo de Dados, Papel/Ação, etc. Modelo de cascata, modelo iterativo, modelo ágil, etc.
Qual é o custo da Engenharia de Software	60% desenvolvimento, 40% Teste. Adiante, mais informações.
O que é CASE (computer-aided software engineering)?	<b>"Engenharia de Software auxiliada por computador".</b> Automação do processo de requisitos, modelagem de sistema, depuração e teste. As vezes até gera o código.

# Perguntas e Respostas - Glossário

O que é o Artefato?	Conjunto de programas, conteúdo (dados), modelos, documentos, o software.
Quais são os desafios-chave da engenharia de software?	Redução do tempo de entrega e desenvolvimento de software digno de confiança.
O que são métodos de engenharia de software?	Abordagens estruturadas para desenvolvimento de software que incluem modelo de sistema, notação, regras, recomendações de projeto e guias de processo.

# Campos de Aplicação de Software:

- Software de Sistema (programa para atender outro programa)
- Software de Aplicação (facilitar operação comercial ou técnica)
- Software Científico (processamento numérico muito pesado)
- Software Embutido (residente em um produto. Medidor de combustível)
- Software para linha de produto (processador de texto, planilha)
- Aplicações para Web
- Aplicações Mobile
- Software de inteligência artificial (algoritmos não numéricos, ex: BOT)
- Software para IOT
- Software aberto

# Mais definições que vamos repetir muito....

prag·má·ti·co

adjetivo

1. Relativo à pragmática ou ao pragmatismo.
2. Que tem motivações relacionadas com a .ação ou com a eficiência. = PRÁTICO

adjetivo e substantivo masculino

3. Que ou quem revela um sentido prático e sabe ou quer agir com eficácia.

prag·ma·tis·mo

(inglês pragmatism)

substantivo masculino

Doutrina que toma por critério da verdade o valor prático e se opõe ao intelectualismo.

MANTRA

PRAGMATISMO



Preocupe-se com seu trabalho

Por que passar sua vida desenvolvendo software se não estiver interessado em fazê-lo bem?

Não tolere janelas quebradas.

Corrija projetos incorretos, decisões erradas e códigos frágeis quando os encontrar.

# Momento Pragmático – Socioemocional

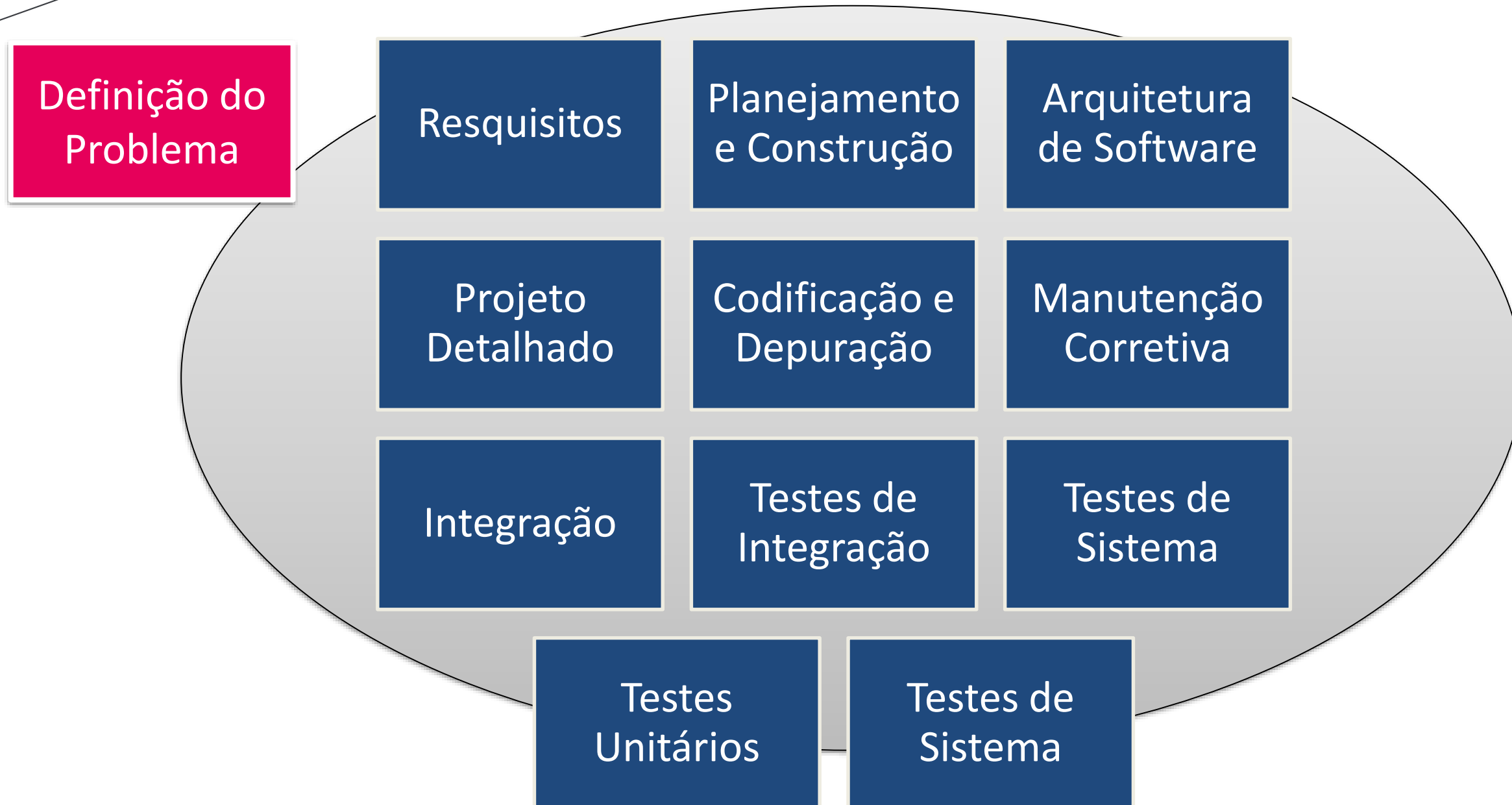
Alguns já estão precisando de uma caçamba...



- Ajude a fortalecer sua equipe examinando as **“vizinhanças” de seu ambiente de computação.** Selecione duas ou três janelas quebradas e discuta com seus colegas quais são os problemas e o que poderia ser feito.
- Você consegue identificar quando uma janela foi quebrada? Qual a sua reação? Se isso foi resultado de uma decisão de alguém, ou de uma imposição da gerência, o que você pode fazer?



# O que é Construção de Software?



---

SWEBOK

---

Requisitos de software

---

Projeto de software

---

Construção de software

---

Teste de software

---

Manutenção de software

---

Gerência de configuração de software

---

Gerência de engenharia de software

---

Processos de Engenharia de Software

---

Ferramentas e Métodos de Engenharia de Software

---

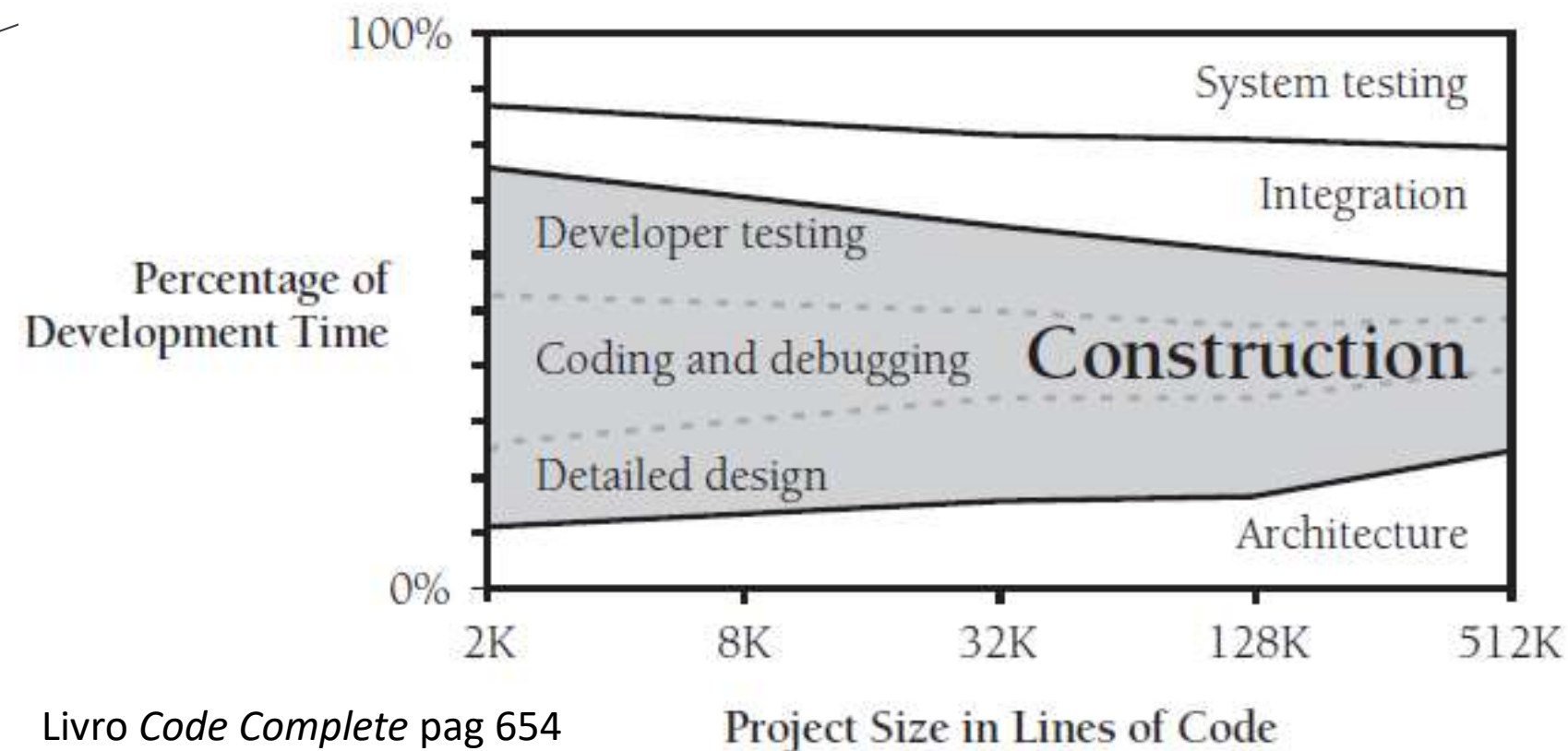
Qualidade de software

---



**IEEE – Instituto  
de Engenheiros,  
Eletricistas e  
Eletrônicos.**

# Por que é importante a Eng. de Software?



Livro *Code Complete* pag 654

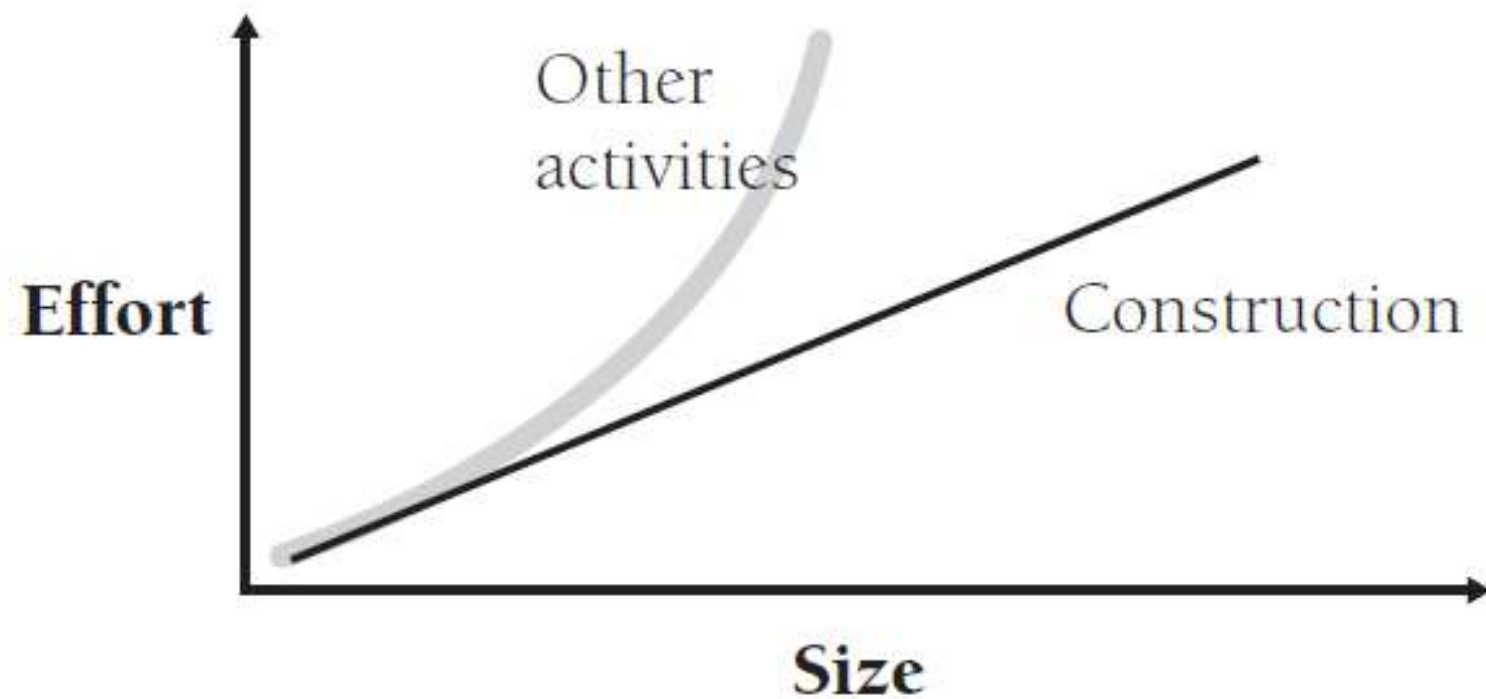
A medida que o tamanho do sistema aumenta, o tempo em nas atividades de Arquitetura, Integração e Teste também aumentam.

! O tempo da construção do software reduz proporcionalmente

Atividades que aumentam:

- Comunicação
- Planejamento
- Gerenciamento
- Lev. Requisitos
- Projeto Funcional
- Arquitetura
- Integração
- Remoção de Defeitos
- Testes
- Documentação

# Por que é importante a Eng. de Software?



**Quando você faz um castelo a chance de esquecer a porta aumenta muito!**



Você recebe o código fonte do jogo “Pokemon Go” para adicionar uma funcionalidade. Ninguém dúvida que é um software legal e que funciona bem, então você abre o fonte e não tem documentação, olha o código e está todo duplicado e indentação passou longe...então do ponto de vista de qualidade de software, ele é bom ou ruim?

**Facilidade de manutenção também é uma resultante de projeto bem feito!**

# Escala de Projetos de Software

Escala	Tamanho em Linhas de Código	Como pode ser feito
Muito pequena	Até 100	Um bom programador pode fazer intuitivamente
Pequena	Até 1.000	Um bom programador consegue fazer com processos informais
Média	Até 10.000	Um bom programador consegue fazer com processos definidos
Grande	Até 100.000	Uma equipe consegue fazer com processos definidos
Muito grande	Até 1.000.000	Uma organização consegue fazer com processos definidos
Múltipla	Mais de 1 milhão	Pode necessitar de um conjunto de organizações cooperantes

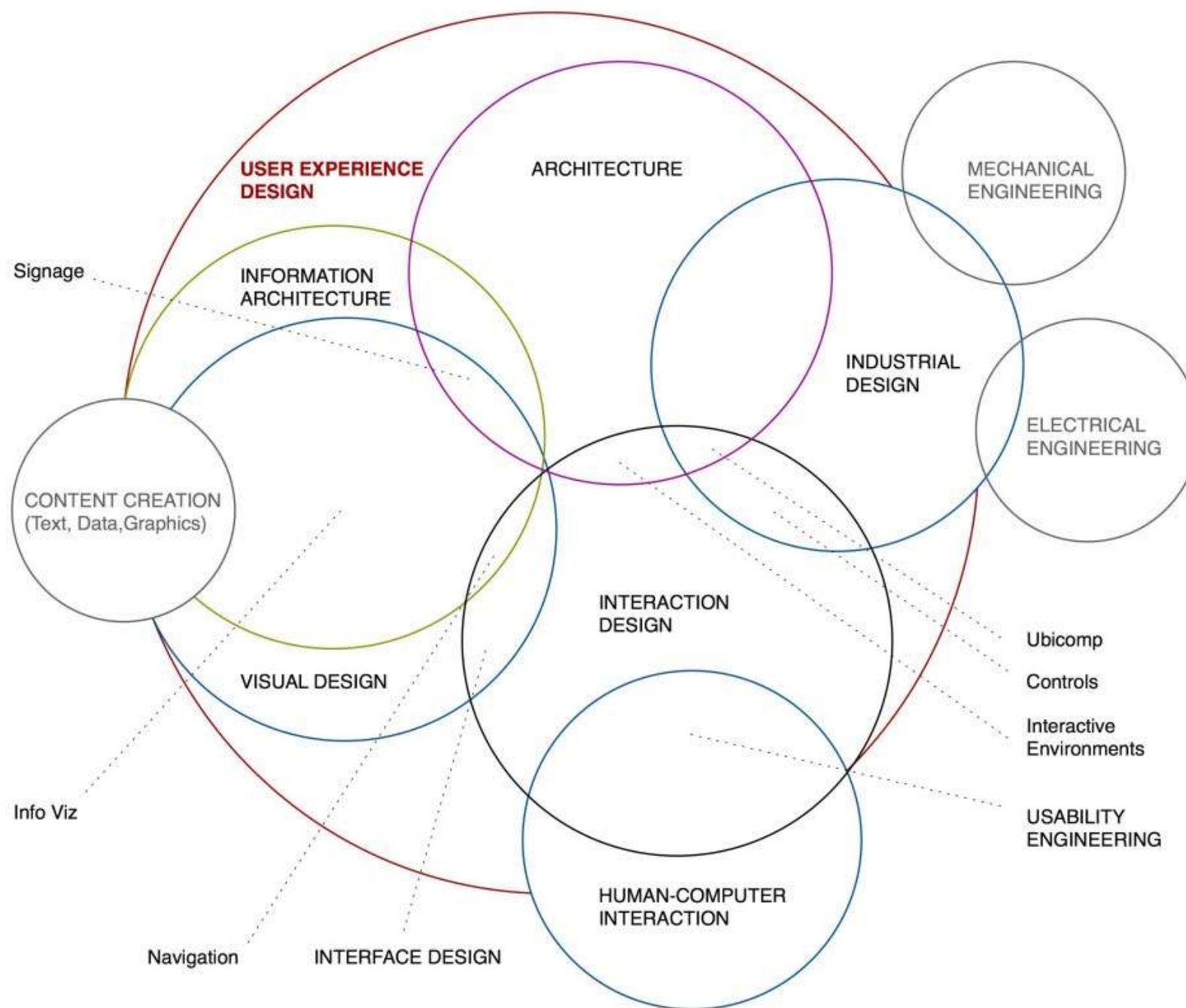
# Complexidade de Comunicação

**Exercício no Quadro, utilizando “bolinhas”....**

**....falando de Gente....vamos para UX....**

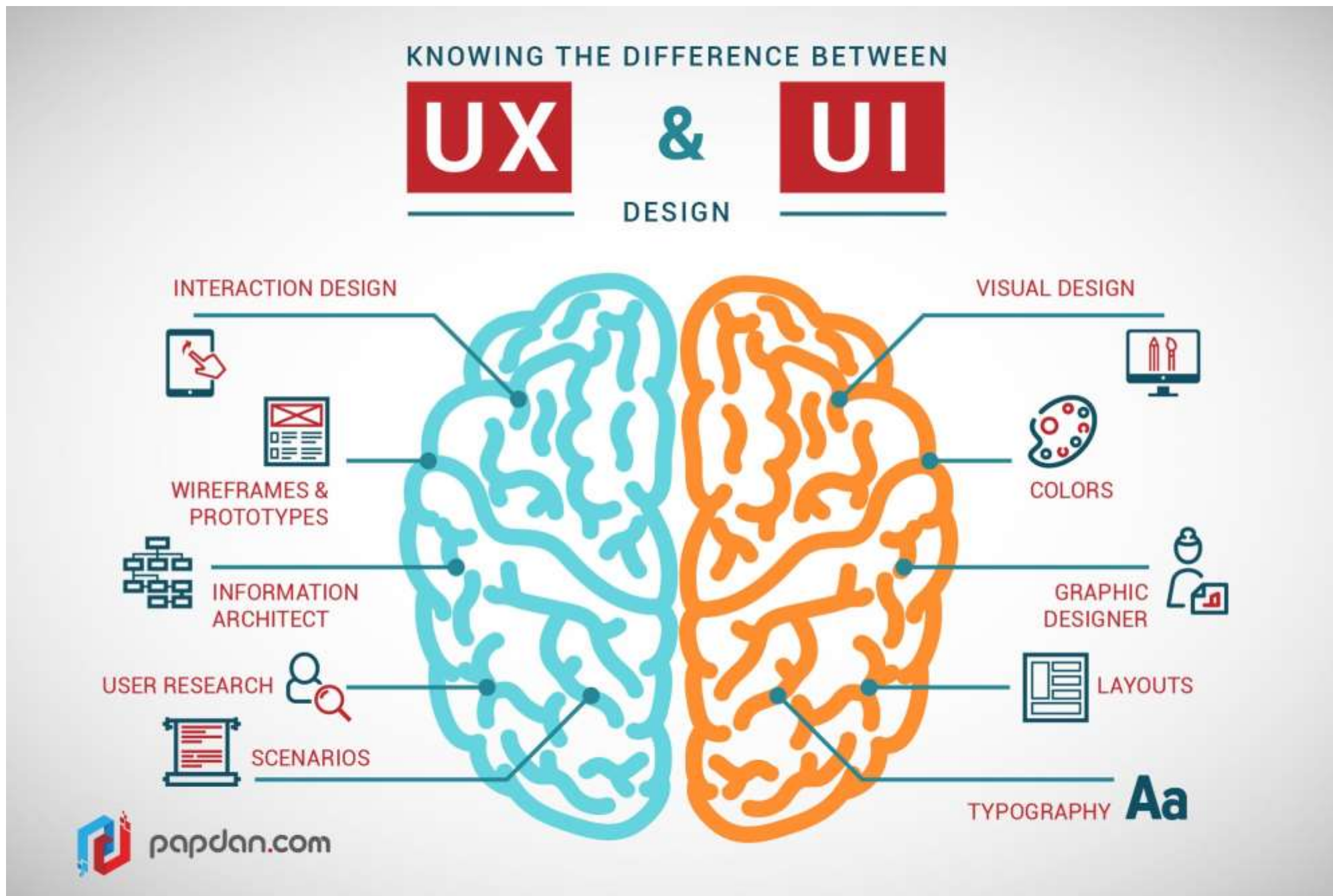


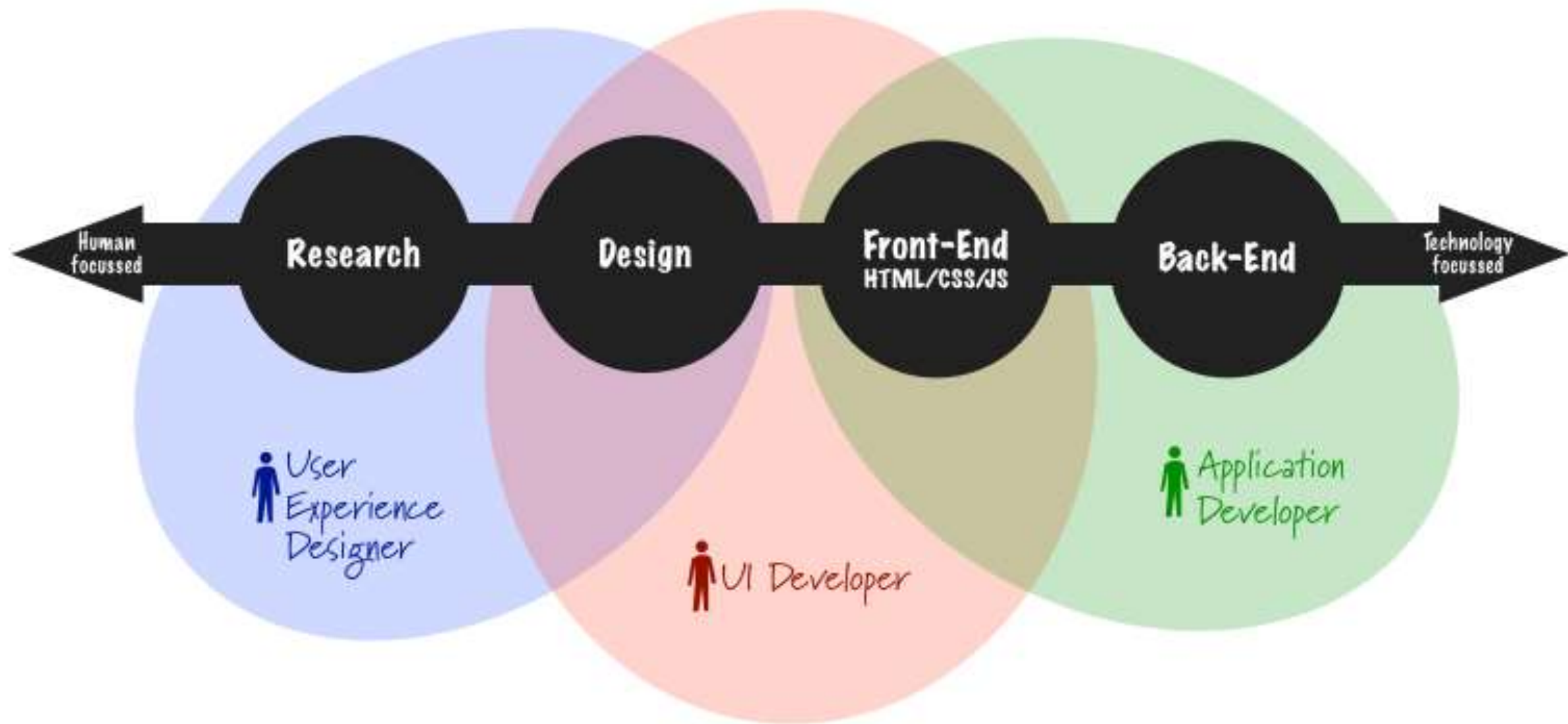
# Disciplinas



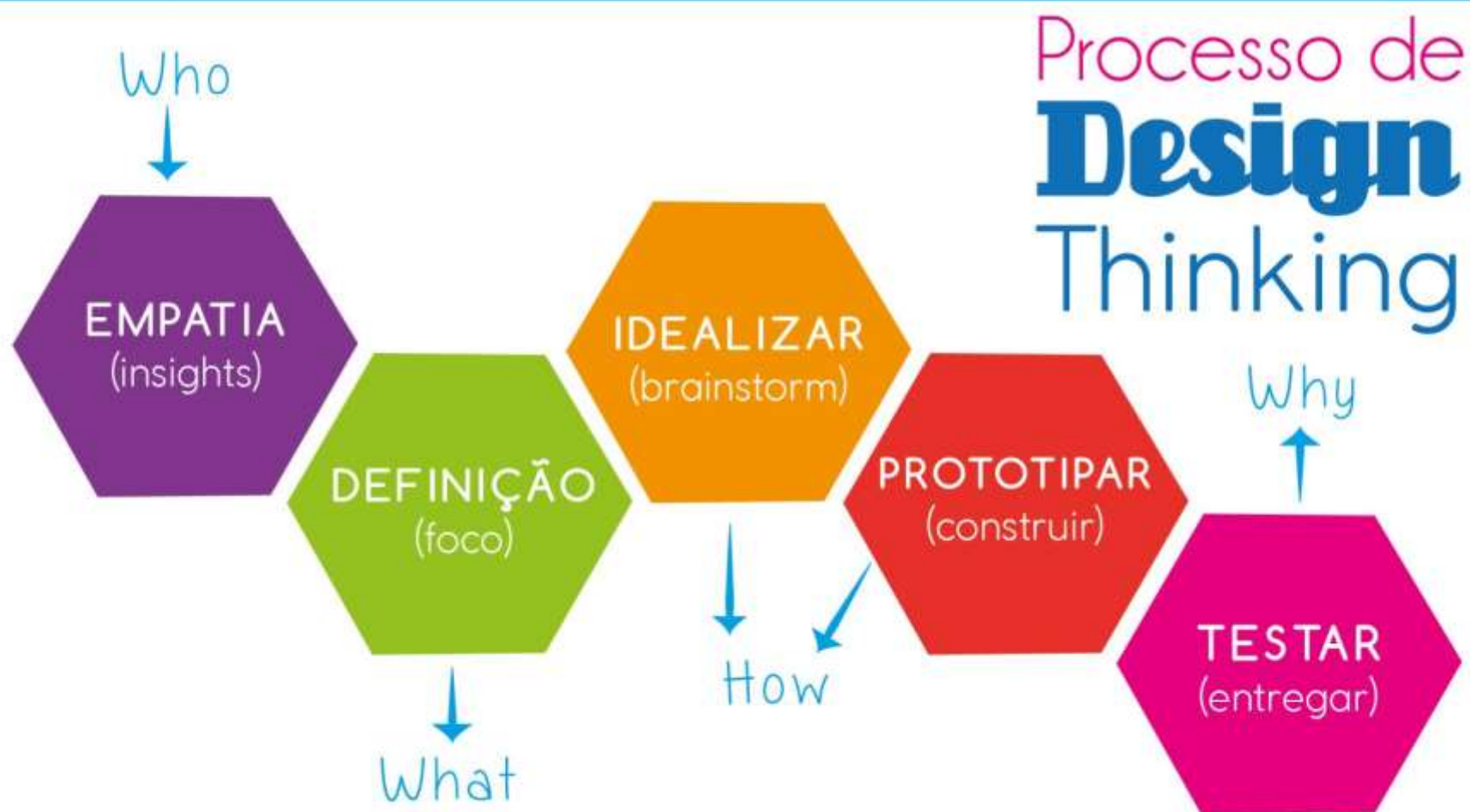


# UX & UI





# Semelhança com Design Thinking



UTILIZADO NORMALMENTE EM PROCESSO DE INOVAÇÃO.  
ENTENDER A REAL NECESSIDADE DO USUÁRIO.

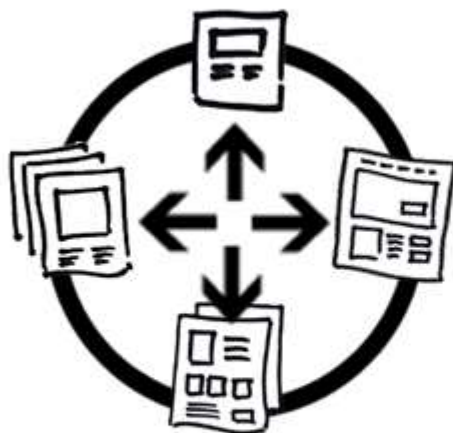
# Olha o Google

dia **1**



entender  
definir

**2**



divergir

**3**



decidir

**4**



prototipar

**5**



validar





# Frase Chave na Sprint 1

Talvez a sua opinião seja importante.

# Palavra Chave da Sprint 1

EMPATIA!

# Objetivo de Estudo – Design de Interação

Humanos, Computadores e os fenômenos das interações entre as duas “espécies”



- O desenvolvimento e melhoria da utilização, utilidade, segurança, eficiência e eficácia dos sistemas;
- A melhoria da usabilidade dos produtos, ou seja, tornar os sistemas mais fáceis de usar e aprender.

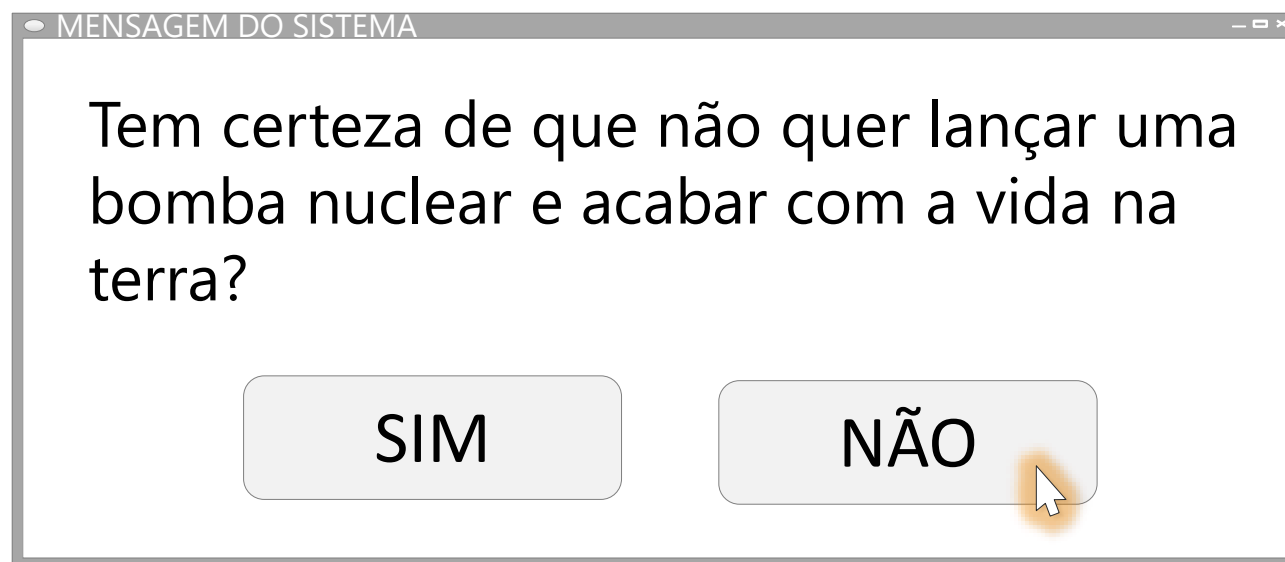
## Para que?

O design com boa experiência gera satisfação, e os ganhos comerciais normalmente são muito maiores que o custo do redesenho.

- Seja possível realizar número maior de tarefas e de forma mais veloz;
- Realização de novas tarefas, cada vez mais rápido;
- Suportar o processo de resolução de problemas dos utilizadores;
- Promover o desempenho e resultados mais confiáveis;
- **Atingir os objetivos comerciais (vender mais, reter mais...)**



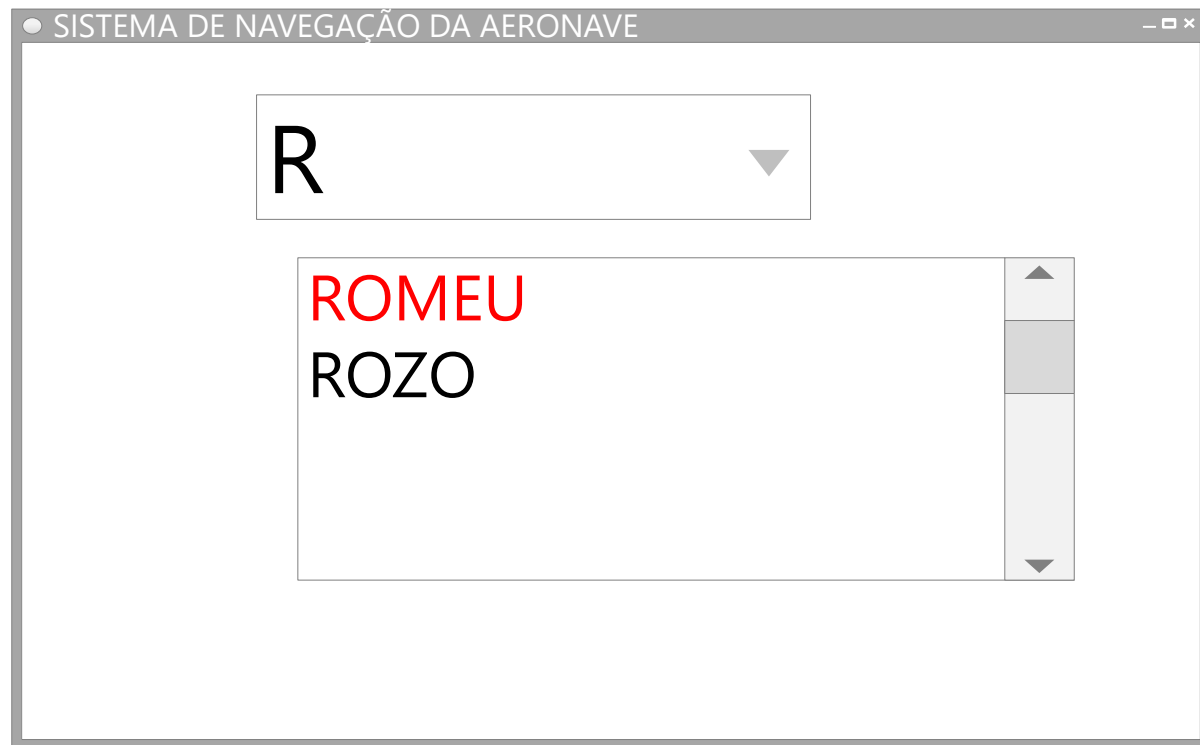
# Exemplo 1 – Mensagens Negativas



A frase acima está exagerada, mas a questão é que frases NEGATIVAS podem oferecer grandes riscos de interpretação.

## Exemplo 2 – Auto completar

Voo 965 – Boeing 757 - Cali



A screenshot of a software window titled "SISTEMA DE NAVEGAÇÃO DA AERONAVE". Inside the window, there is a text input field containing the letter "R". Below the input field, a dropdown menu is open, displaying two options: "ROMEU" in red text and "ROZO" in black text. The dropdown menu has a vertical scrollbar on its right side.

O avião precisava seguir a rota ROZO, mas o piloto foi induzido para outra rota e a aeronave se chocou com a montanha.

## Exemplo 3 – THERAC-25

### Mensagem de Erros e Controles inapropriados no Raio-X

➤ ANTES

CONTROLE DE DO RAIO X

14848

7 8 9

4 5 6

1 2 3

0

➤ DEPOIS

CONTROLE DE DO RAIO X

1 4 8 4

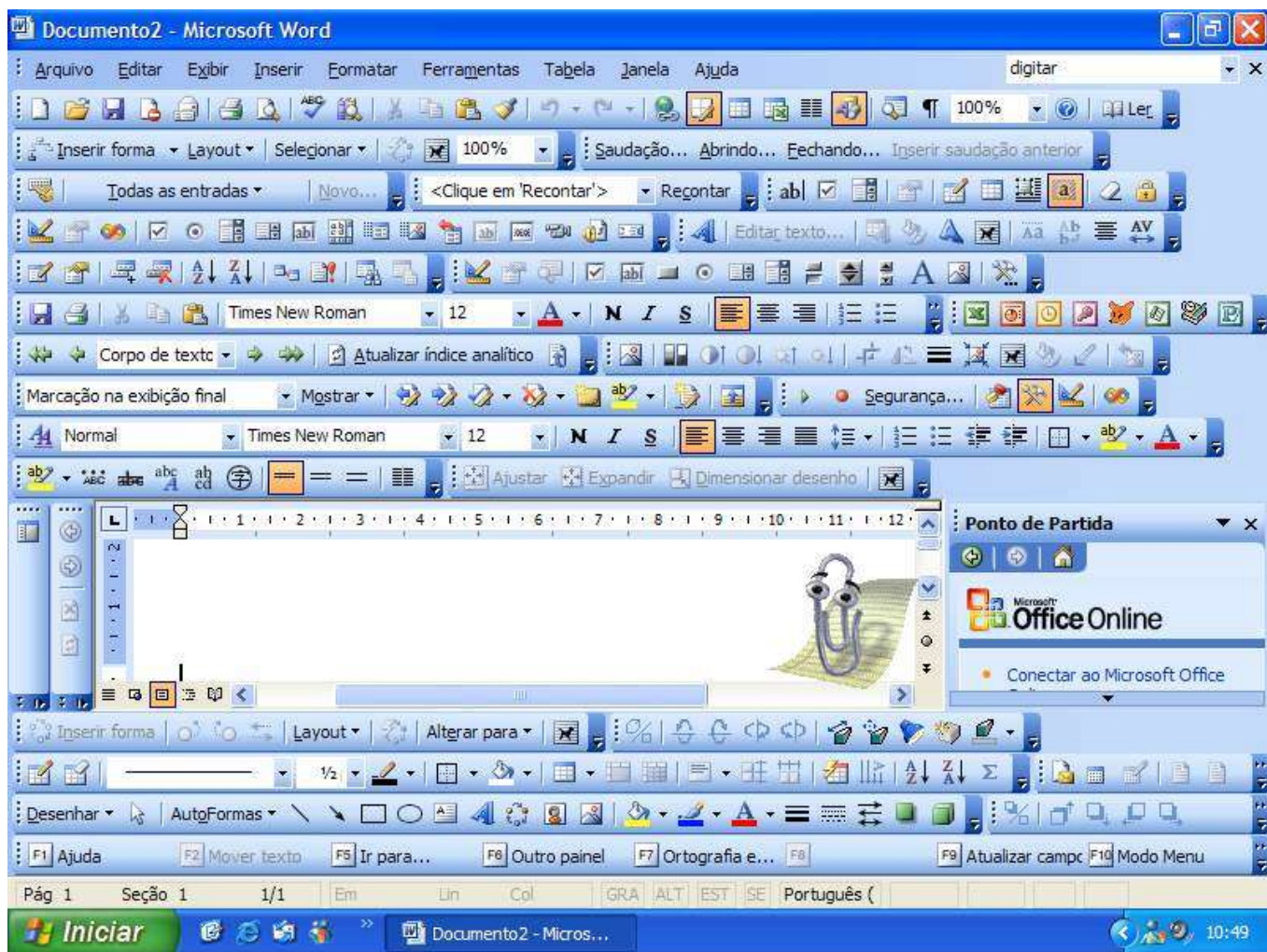
+ + + +

- - - -

Erro de digitação, o sistema apresentou o erro “54” e na interface apareceu “sub dose”. O procedimento foi repetido várias vezes e meses depois o paciente veio a óbito.

## Exemplo 4 – Funcionalidade Aguda

“creeping feature”. Fabricantes de software anunciam seus produtos pelas listas de funcionalidades.



- Você já usou todas essas funções do Word?
- Quanto custa manter tudo isso?
- Corrigir os bugs?
- Evoluir?
- Alguma semelhança com as propagandas de site?

➤ Homenagem póstuma ao Clippy, assistente do Office 1997-2003. Prenúncio do Bot?

## Exemplo 4 – Operadora de Telefonia

Interface não é apenas Tela, teclado e mouse

➤ ANTES

“Bom dia! Para conserto tecle 1, contas 2...”

➤ DEPOIS

“Bom dia! Meu nome é João, seu atendente virtual. Consultei aqui e notei que existe problema com o seu equipamento, o senhor deseja agendar um reparo? Digite 2 para sim <pausa> e 3 para não.”

Clientes solucionando seu problema e economia de milhares de reais.

# Metas de Interface

- 😊 Útil
- 😊 Satisfatório
- 😊 Agradável
- 😊 Atraente (engaging)
- 😊 Prazeroso (pleasure)
- 😊 Emocionante/Excitante
- 😊 Interessante (entertaining)
- 😊 Humanizado
- 😊 Motivador
- 😊 Desafiador
- 😊 Social
- 😊 Divertido
- 😊 Recompensador

- 😞 Não cumpre/Não resolve
- 😞 Tediioso
- 😞 Faz você se sentir culpado
- 😞 Irritante
- 😞 Infantil
- 😞 Desprazeroso
- 😞 Faz você se sentir estúpido
- 😞 Condescendente (padronizing)
- 😞 Forçosamente bonito (cutesy)
- 😞 Artificial/falso
- 😞 Inconsistente

- Visibilidade  
Ex: Ícone de msg que chegou, controles dos carros.
- Feedback (tem relação com visibilidade)  
Ex: Joystick que vibra, tela que treme, mensagem de alerta.
- Restrições  
Ex: Máscara em campo texto, botões desativados, travas.
- Consistência  
Ex: Uso de convenções como local dos controles ou tipos de ícones.
- Affordance  
Ex: Dar um “pista” de como usar, o famoso, “intuitivo”.



## Tarefa em dupla (Grupos de PI)

- Pesquise um ou Website que seja concorrente ou similar ao projeto de vocês.(pode ter: WebChat, BOT, App)
- Analise o site, liste as metas de interfaces com breve explicação (boas ou ruins)
- Olhe os princípios de design e sugira mudanças, que você provavelmente vai fazer no seu 😊

<https://www.lingscars.com/>



<https://vidaprogramador.com.br/2015/04/23/modulo-de-download/>