

# Verificação de Software

Rodrigo Ribeiro  
Ouro Preto, 19/01/2018

# Quem sou eu?

- Professor da UFOP desde 08/2008
  - Departamento de Computação e sistemas: 08/2008-07/2017
  - Departamento de Computação: 08/2017-...
- Ex-aluno do DECOM/UFOP
  - Graduação em 2005
  - Mestrado e Doutorado pela UFMG.

# Pesquisa

- Temas principais:
  - Projeto de linguagens de programação
  - Engenharia de software.
- Foco:
  - Como construir software correto?

Quando podemos  
afirmar que um  
software é correto?

# Correção de Software

- Um software é correto se ele atende os seus requisitos
  - Ou seja, quando ele faz o que deveria fazer.

Algum exemplo de  
software correto?

# A Sociedade diz que...



# A Sociedade diz que...



Não existe essa  
história de software  
correto!

**Na história da Computação,  
existem diversas falhas  
catastróficas  
causadas por software**

Mas, devemos nos preocupar com  
correção de software?



**SHOULD  
WE BE  
WORRIED?**

# Software is everywhere

- Sistemas bancários
- Aparelhos médicos
- Automóveis
- Celulares
- Aviões
- TV's
- Automação residencial...



# Devemos nos preocupar?

- Software faz parte de nosso dia-a-dia.
- Erros de software podem causar:
  - Prejuízos financeiros
  - Risco de vida
- Pergunta: você entraria em uma viagem em um carro robô na BR-381?



Yes!!!!!!

Algumas falhas causadas  
por software...

# Toyota Prius

- Primeiro híbrido produzido em massa.
- Fev. 2010: Erro no sistema de freio ABS.
  - Causou vários acidentes.
  - Recall de 185.000 veículos.



# Foguete Ariane 5

- Foguete da European Spacial Agency.
- 04/06/1996: Explode 37 seg. após lançamento.
- Causa: Erro em um cálculo no software embarcado no foguete.
  - Conversão de um númer. de 64 bits para um de 16 bits.



# London Ambulance Service

- Equivalente ao SAMU.
- Entrou em operação em 1992.
- Erro de cálculo de posição
  - Envio de ambulâncias a localidades incorretas.
  - Vários veículos a mesma localização.
  - Ocasionalmente a morte de cerca de 30 pessoas.



# Windows 98

- Apresentação de lançamento do Windows 98: blue screen of death.
- Motivo: problemas de alocação de memória.

Name	Dll Base	DateStamp - Name
ntoskrnl.exe	80010000	33247f80 nai.dll
atapi.sys	80007000	33248040 SIPORT.SY
Disk.sys	801db000	336015c0 MASS2.SY
Ntfs.sys	80237000	344eeb40 uwvid.sy
NTice.sys	f1f48000	31ec6c80 floppy.SY
Cdrom.SYS	f228c000	31ec6c90 null.SYS
KSecDD.SYS	f2290000	335e0000 .SYS
win32k.sys	fe0c2000	34000000 .dll
Cdfs.SYS	fdca2000	33000000 .sys
nbf.sys	fdc35000	31000000 .a
netbt.sys	f1f68000	31000000 .s
ard.sys	f2008000	31000000 .s
Parport.SYS	fdc14000	31000000 .v
ppdrv.sys	f1dd0000	



# Therac-25

- Equipamento de radioterapia.
- Erros de software ocasionaram a aplicação de dosagens incorretas, ocasionando a morte de 3 pacientes.



# Veículo autônomo

- Falha no software de controle faz veículo autônomo atingir uma apresentadora ao vivo



# Aeroporto de Denver

- Sistema completamente automatizado de bagagem.
- Erros de projeto ocasionaram atrasos e um custo adicional de US\$ 560.000.000,00 por um sistema que não funciona corretamente.



# Falhas de segurança

# Spectre e Meltdown

- Caso recente: Spectre e Meltdown.
- Falha de segurança presente em TODOS os processadores modernos.
- Explora a execução especulativa para invadir o espaço de memória de outros programas.
- Pode ser usado para roubar senhas, fotos, documentos...



# Heartbleed

- Falha de segurança que permite acessar informação protegida pelos protocolos criptográficos SSL/TLS usados na OpenSSL.
- Descoberto por um time de engenheiros do Google.
- Correção afetou diversos sistemas operacionais.



# Criptomoedas

- Criptomoedas e blockchains são uma promissora nova tecnologia.
- Porém, extremamente sujeitas a falhas de segurança.
- Recentemente, US\$ 100.000.000,00 foram congelados por uma falha em um smart-contract Ethereum.



# Code Injection

- Uso de entradas não devidamente verificadas para invadir sistemas.
- Ataque comum em sistemas web.
- Vários tipos de ataque.
  - Cross-site scripting: XSS
  - SQL-injection

## SQL Injection.

User-Id: itswadesh  
Password: newpassword

```
select * from Users where user_id= 'itswadesh'  
and password = ' newpassword '
```

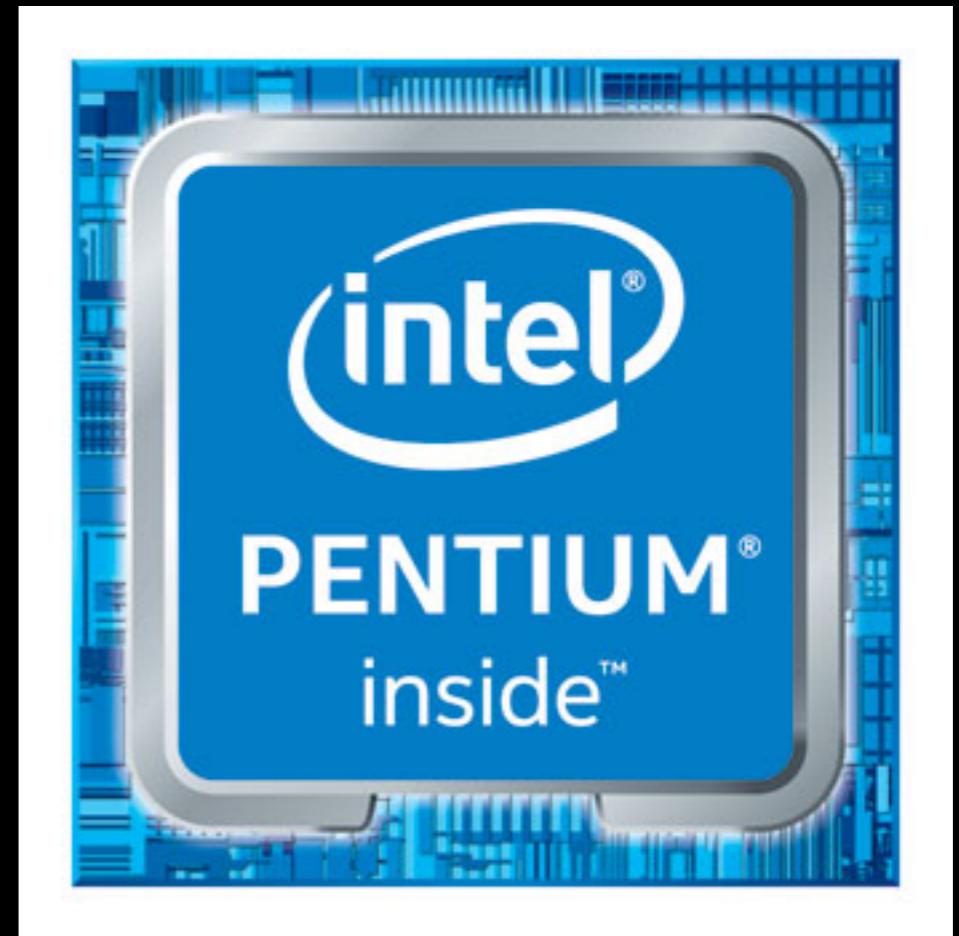
User-Id: ' OR 1= 1; /\*  
Password: \*/--

```
select * from Users where user_id= '' OR 1 = 1; /*'  
and password = ' */--'
```

Somente software  
falha?

# Intel Hardware Bugs

- Bug na unidade de ponto flutuante.
- Erros percebidos após a oitava casa decimal.
- Prejuízos para diversas instituições financeiras.
- Diversos processos sobre a Intel.



# Como verificar software?

# Testes...



# Teste de software

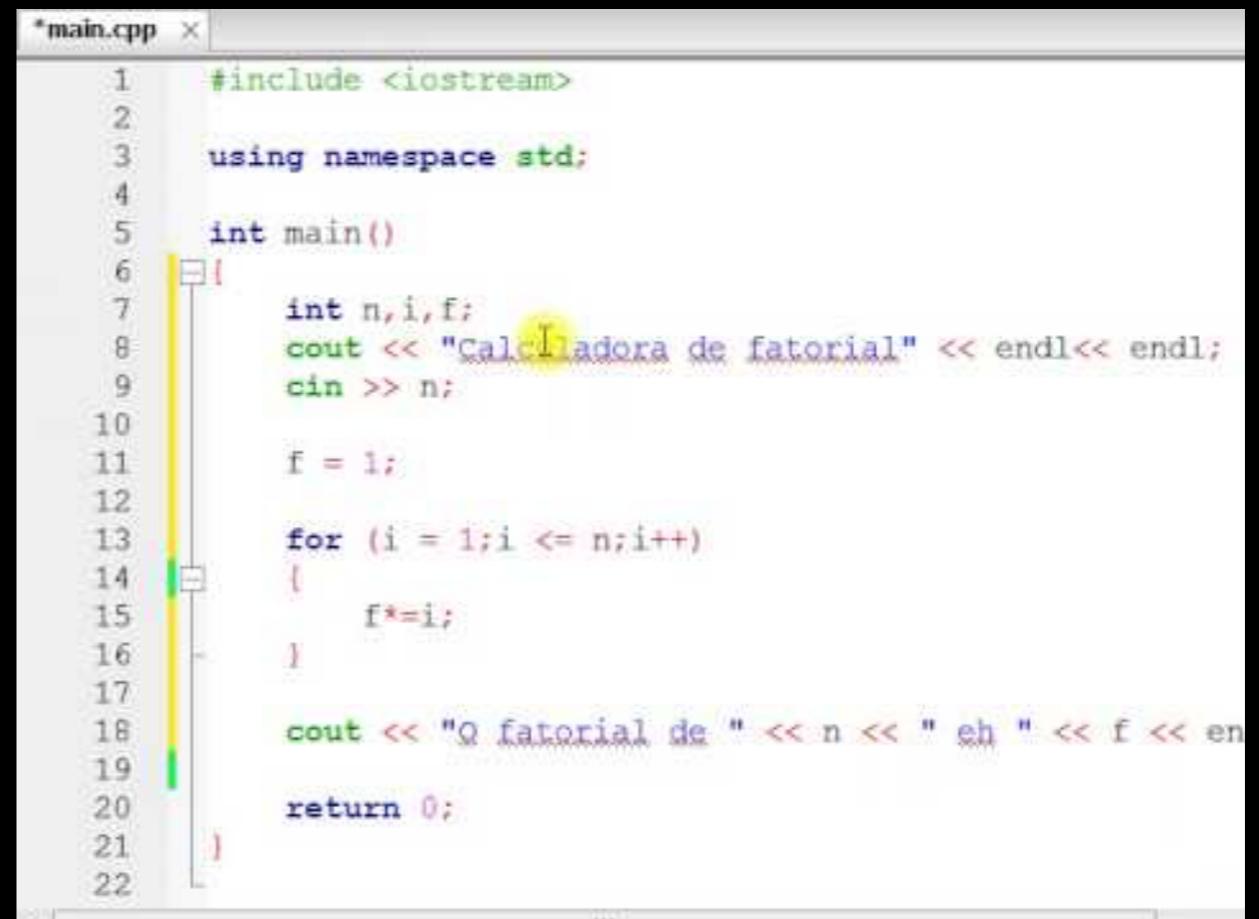
“Tests can only show the presence, not the absence of bugs”.

Edsger W. Dijkstra



# O Problema de Testes

- Para mostrar que um programa é correto, usando testes, temos que testá-lo para todos os possíveis valores de entrada.
- Cada variável inteira tem cerca de  $2^{32}$  possíveis valores.
- Com 3 variáveis, o total de possibilidades a serem verificadas é de  $3 \cdot 2^{32}$



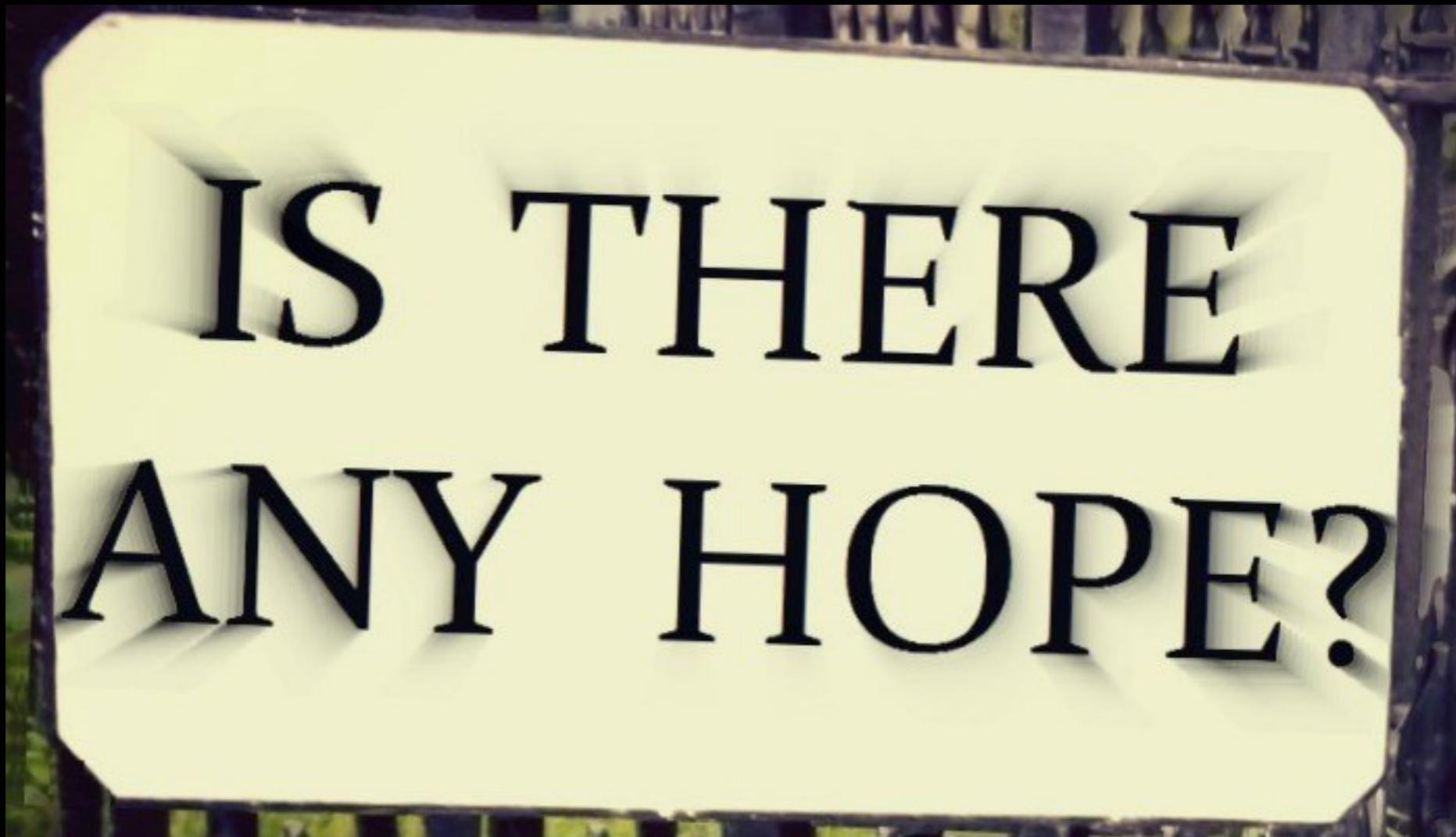
```
*main.cpp x
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int n,i,f;
8     cout << "Calculadora de fatorial" << endl << endl;
9     cin >> n;
10
11    f = 1;
12
13    for (i = 1;i <= n;i++)
14    {
15        f*=i;
16    }
17
18    cout << "O fatorial de " << n << " eh " << f << endl;
19
20
21
22 }
```

A screenshot of a code editor window titled "main.cpp x". The code is a C++ program that calculates the factorial of a number. It includes an include directive for *<iostream>*, a using directive for the *std* namespace, and a main function. Inside the main function, it prints a welcome message, reads a value from standard input, initializes a variable *f* to 1, and then uses a for loop to calculate the factorial by multiplying *f* by each integer from 1 to *n*. Finally, it prints the result back to standard output. The code editor shows syntax highlighting for keywords, comments, and strings.

# Resumindo...



# Alguma Esperança?



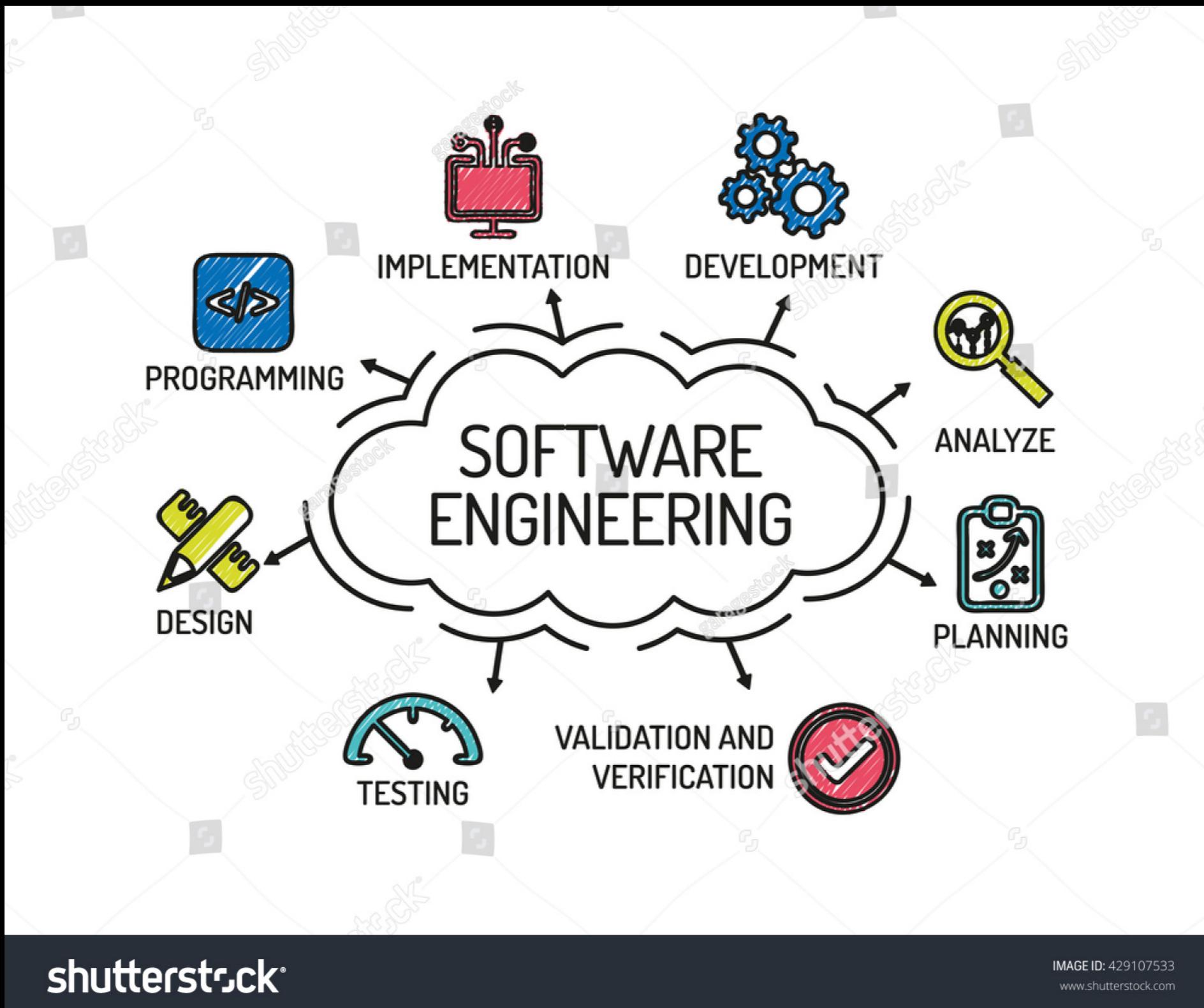
# Engenharias Tradicionais



# Engenharias Tradicionais

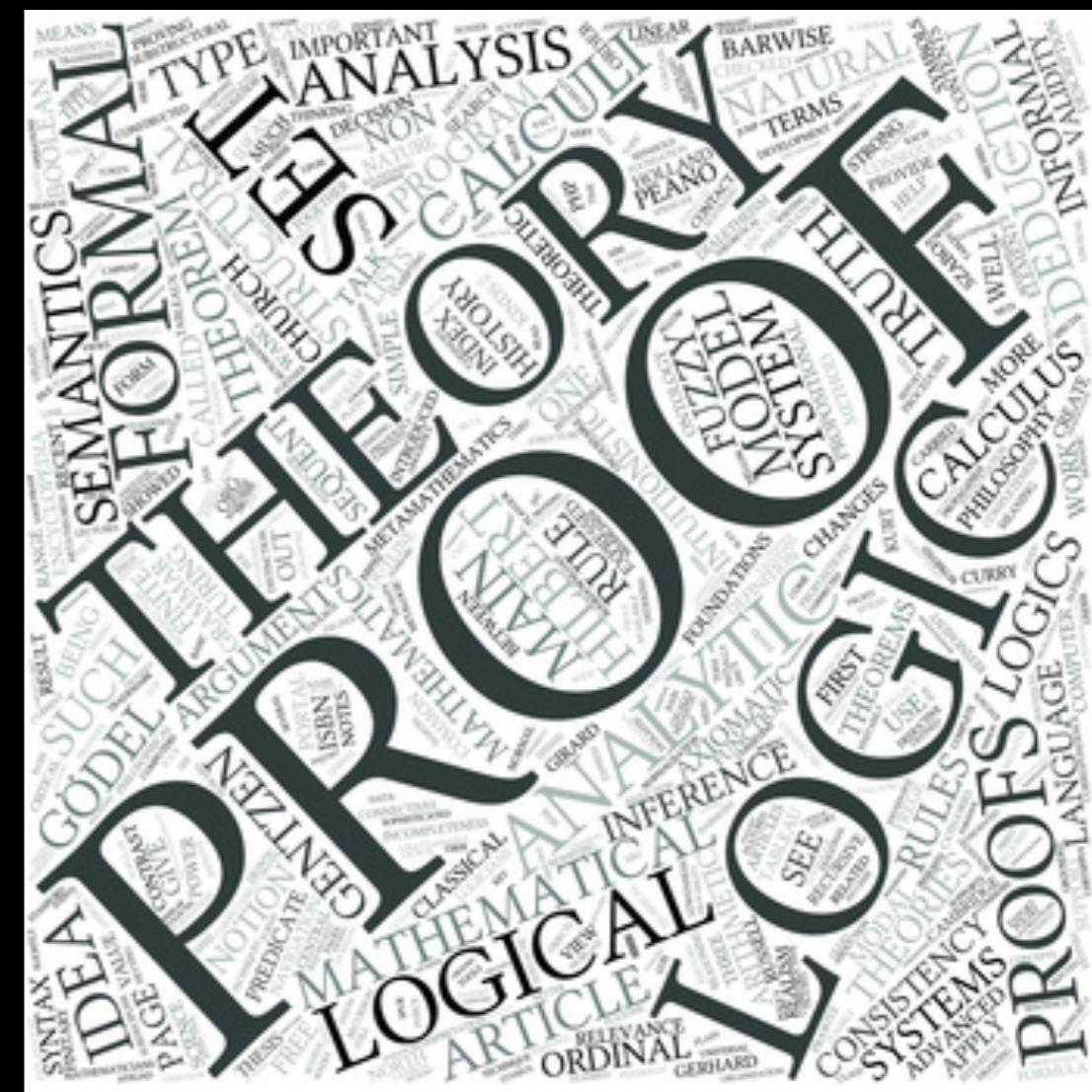
- Ampla experiência: Construção civil existe há mais de 5.000 anos.
- Forte fundamentação teórica
  - Cálculo diferencial e integral.

# Engenharia de Software?



# Engenharia de Software?

- Existe fundamentação matemática para a Engenharia de software?
  - Sim! Lógica formal!

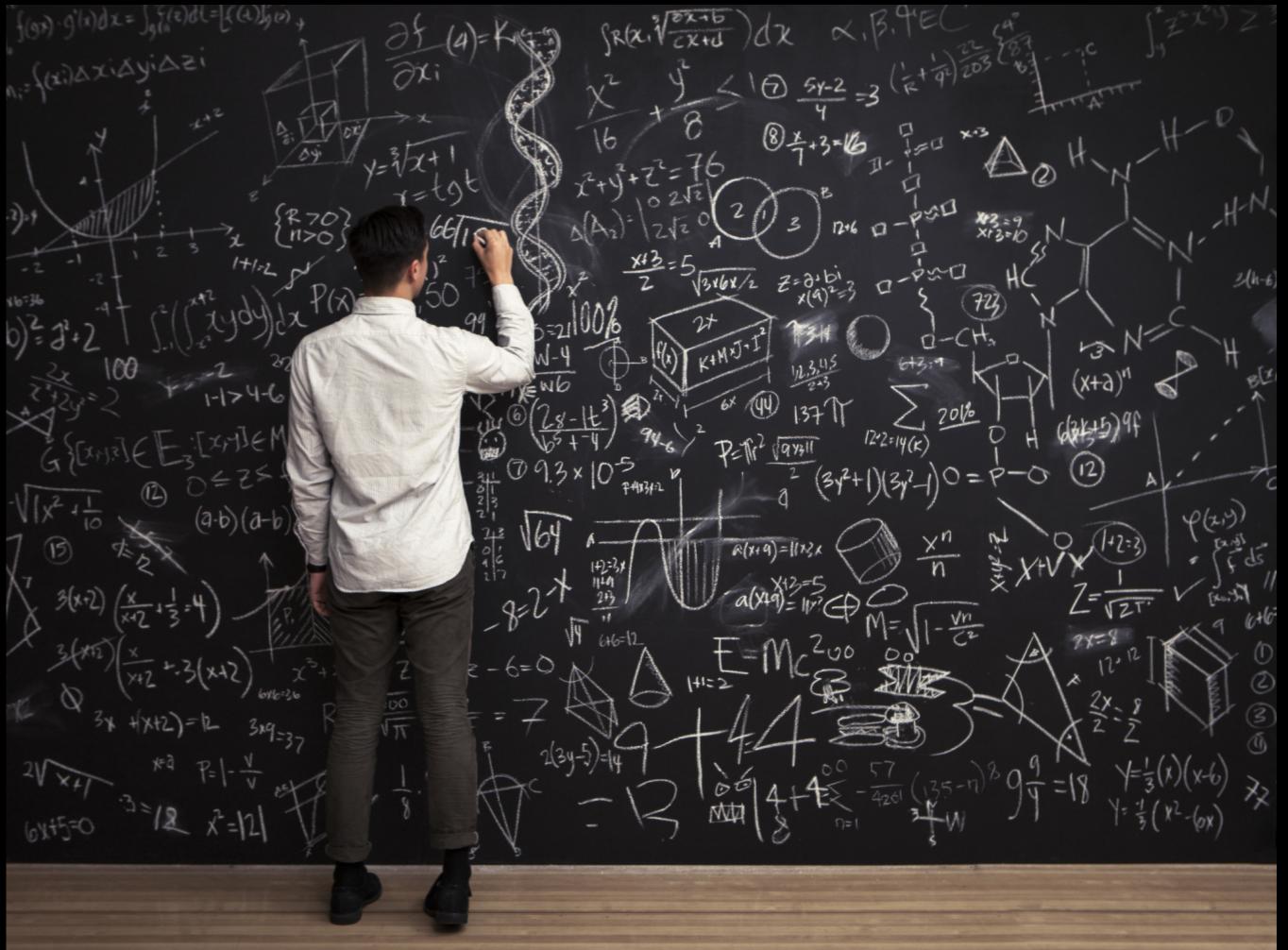


# Lógica Formal

- Usando lógica formal, podemos especificar de maneira precisa os requisitos de um software.
- Tarefa de verificação consiste em uma prova de que o software em questão atende os requisitos.
- Então, porquê não usamos isso?

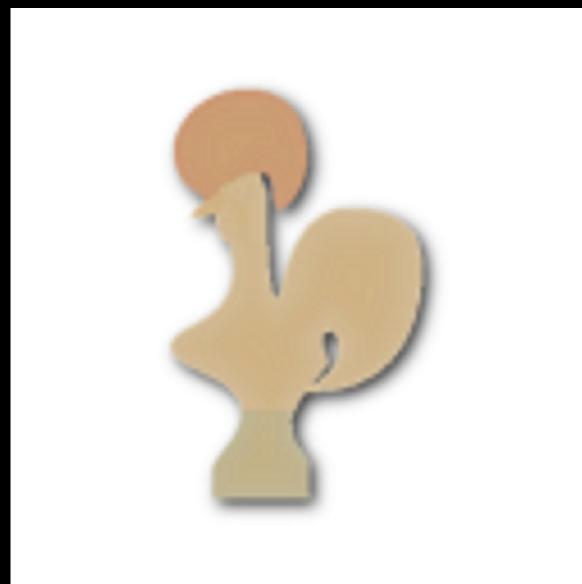
# Lógica e Especificação

- Especificar software é difícil...
    - Os requisitos são consistentes?
    - Existência de ferramentas e profissionais capazes de usá-las.



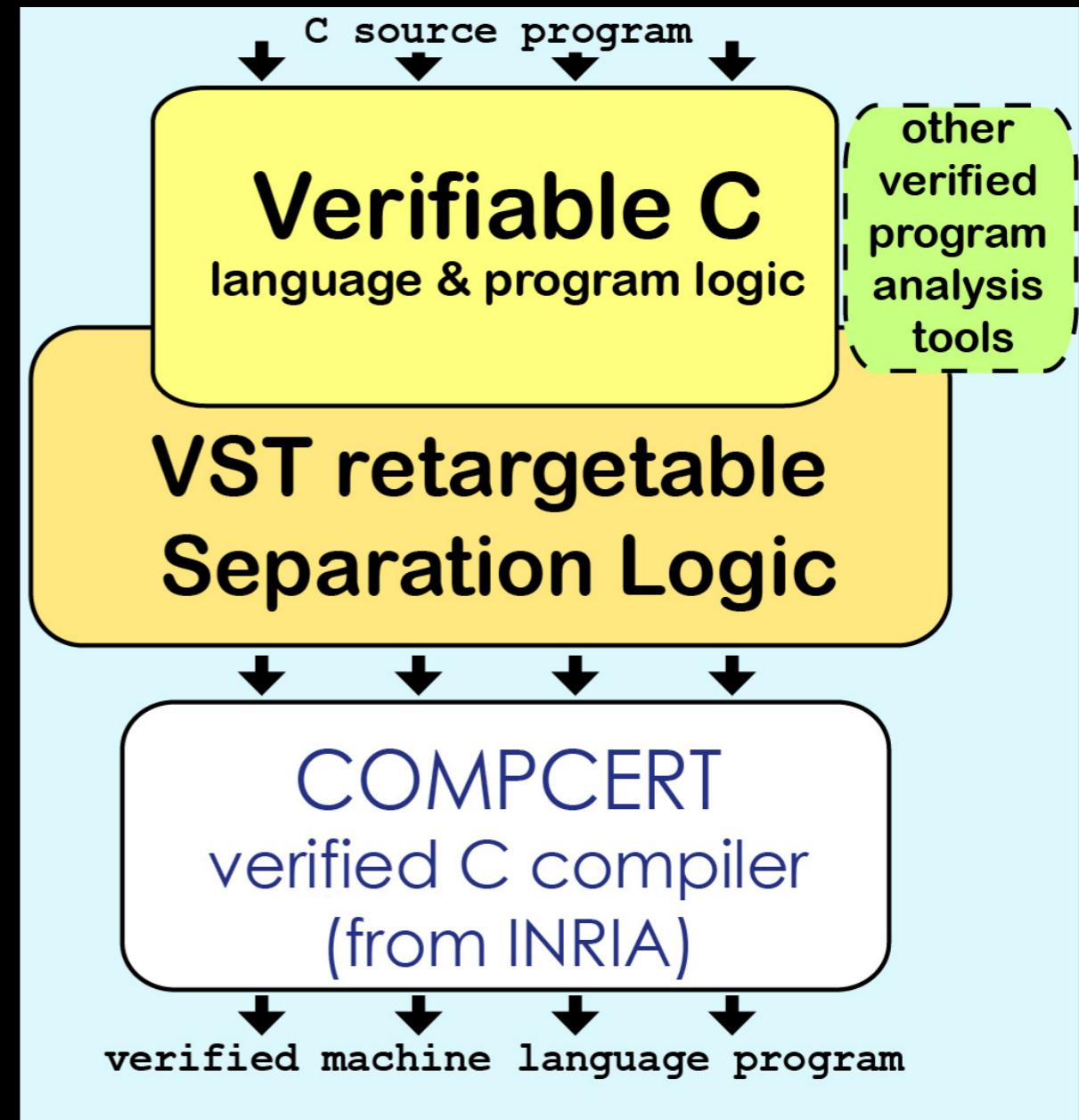
# Lógica e Desenvolvimento

- É possível desenvolver programas corretos a priori?
- Atualmente, existem linguagens que permitem que expressarmos propriedades como tipos de programas.
- Verificação feita pelo compilador.



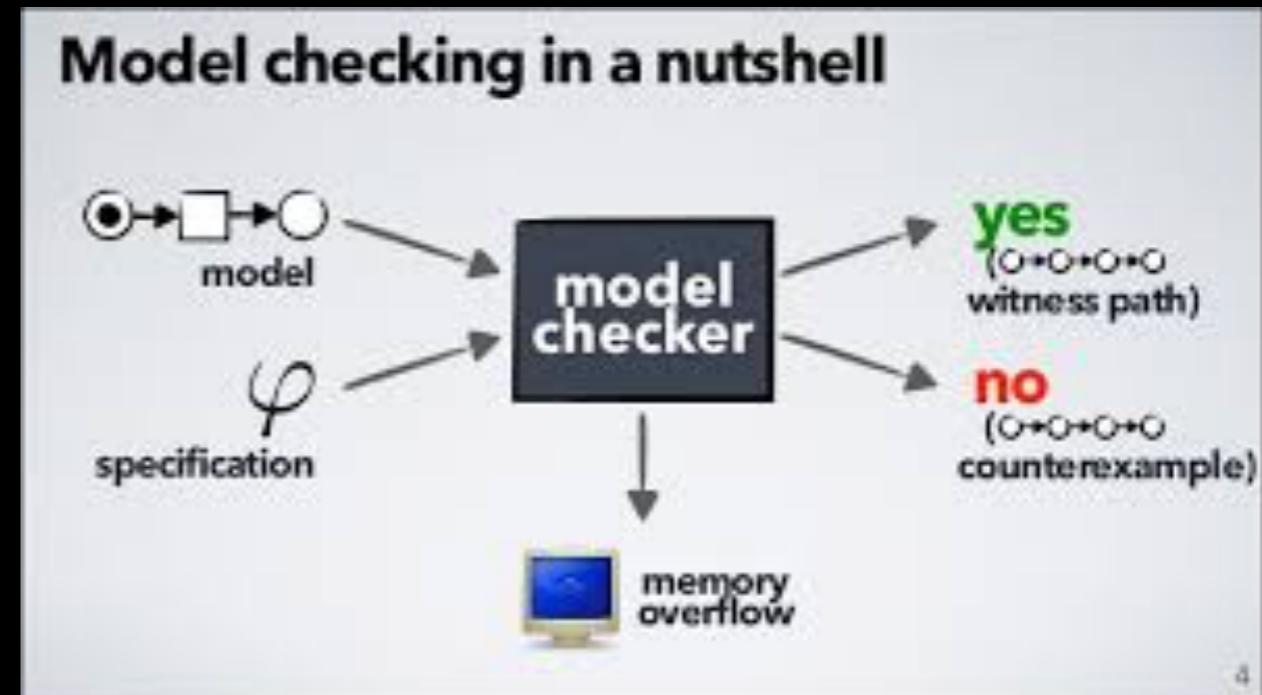
# Lógica e Validação

- Validação dedutiva:
  - Uso de lógicas específicas para verificar programas.
  - Vantagem: pode-se verificar qualquer programa.
  - Desvantagem: requer especialistas.



# Lógica e Validação

- Verificação de modelos
  - Usa um modelo do software e o verifica com respeito a uma fórmula de especificação.
  - Vantagem: automático
  - Desvantagem: limitado para software. Muito usado para hardware



# Mas então...

- Verificar software é muito difícil!
- Soluções atuais não são fáceis de usar ou são limitadas.



# Oportunidades

- Verificar software é difícil.
  - Soluções atuais não são ideais.
  - Isso significa oportunidades de pesquisa!



# Alguns Problemas...

- Desenvolvimento de sistemas embarcados corretos por construção.
- Verificação de smart-contracts.
- Verificação de algoritmos de parsing.
- Projeto e implementação de linguagens de programação



# Sistemas Embarcados

- Programar sistemas embarcados em linguagens de alto nível.
  - Atualmente: C/C++
- Como testar sistemas embarcados de maneira automatizada?
- Desenvolvimento de uma plataforma para verificação



# Smart-contracts

- Software capaz de realizar transações financeiras sem a necessidade de intermediários.
- Podem ser usados para:
  - Pagamentos.
  - Gerenciamento de benefícios (milhas, pontos de cartão de crédito, ...)
  - Jogos (criptokitties)



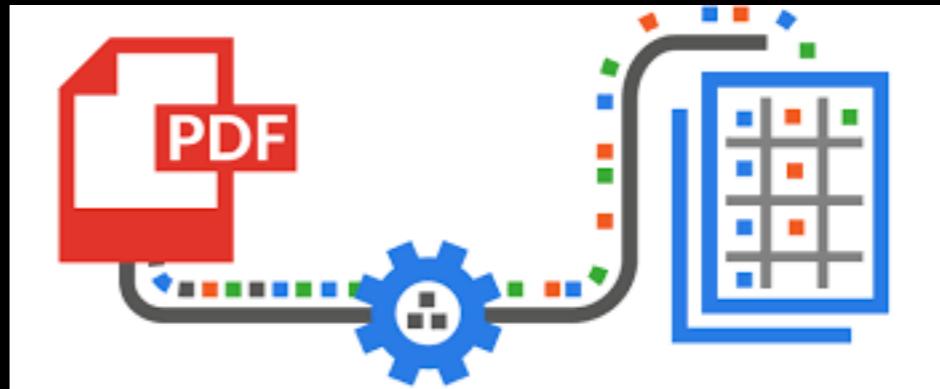
# Oportunidades

- Como testar smart-contracts de forma automatizada?
- Como verificar propriedades de interesse?
- Como garantir que a blockchain usada não possui falhas de segurança?
- O que seria uma boa linguagem para desenvolvimento de smart-contracts?



# Algoritmos de Parsing

- Parsing: Analisar uma string de acordo de um formato.
  - Ex: 359000-000 representa um CEP?
- Parsing everywhere:
  - Browsers, protocol de rede, arquivos de imagens, pdf, device drivers...



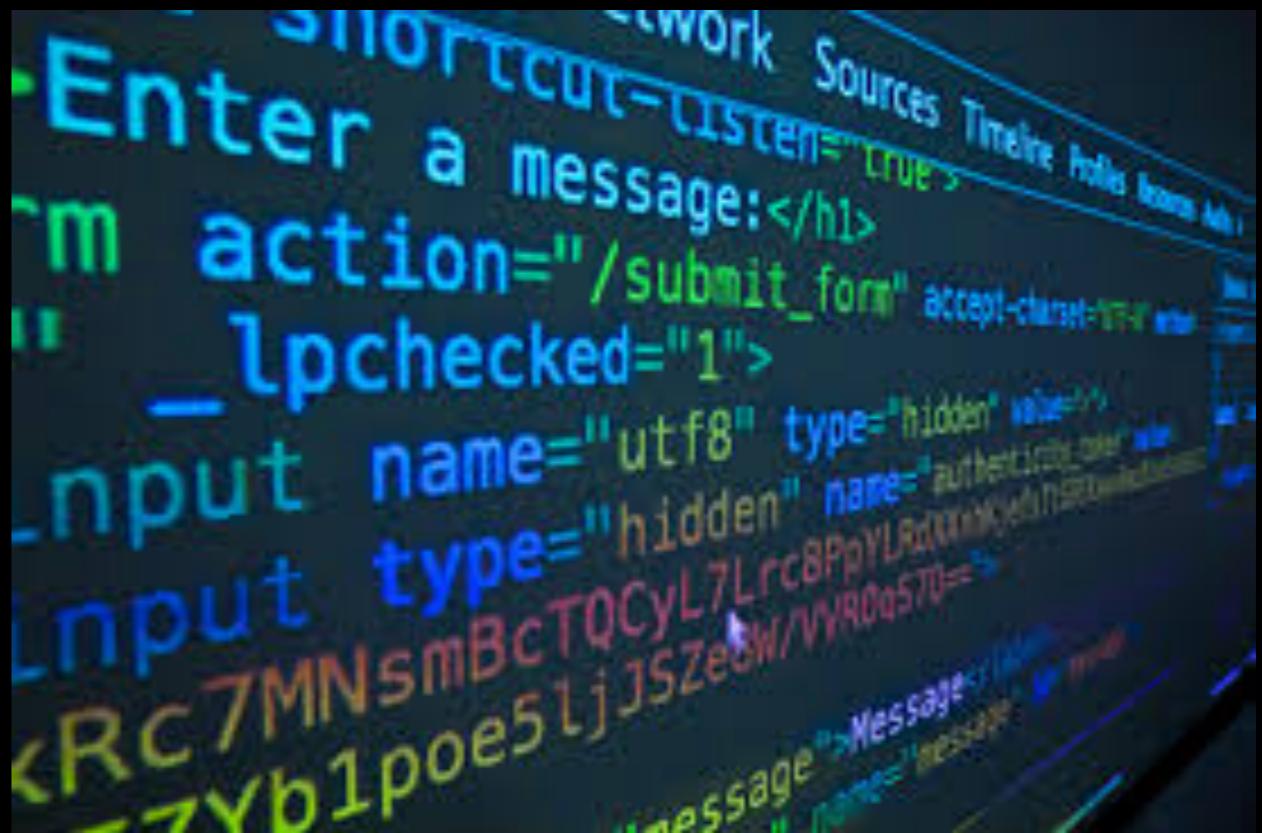
# Oportunidades

- Verificação formal de algoritmos clássicos de parsing.
- Desenvolvimento de ferramentas para testes de segurança.
- Desenvolvimento de novas técnicas para especificação e construção de algoritmos de parsing.



# Linguagens de Programação

- Ferramenta de trabalho de profissionais de computação.
- Como garantir que sua linguagem preferida funciona?
- Seu compilador é correto?
- É possível implementar um compilador correto para sua linguagem?
  - No caso de Java, não é... ;)



# Disciplinas do Curso

- Algoritmos e Estruturas de dados I e II
- Projeto e análise de algoritmos
- Matemática Discreta I e II
- Teoria dos Grafos
- Programação Funcional.
- Teoria da Computação
- Compiladores
- Lógica aplicada à computação

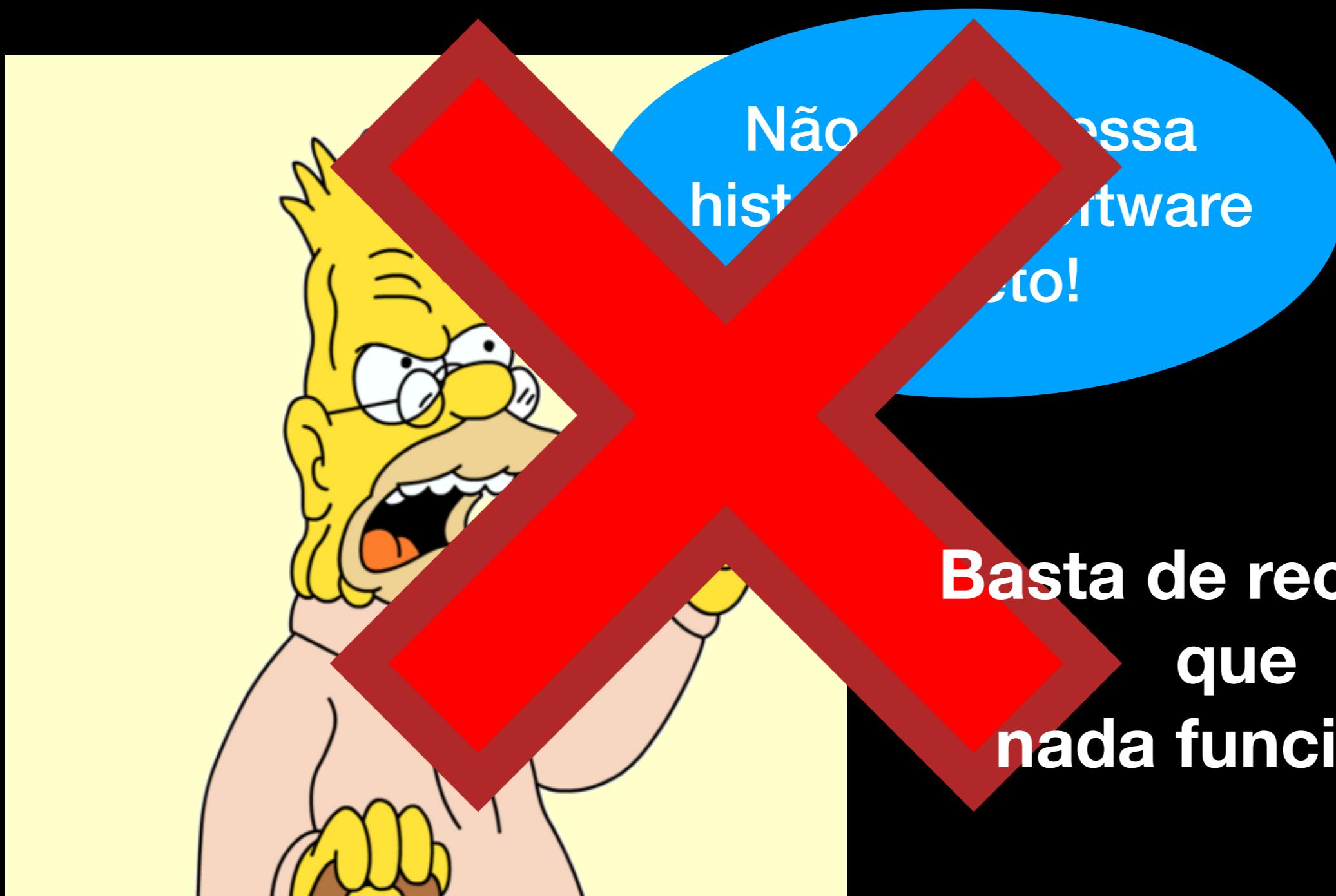


# A verdade...



Não existe essa  
história de software  
correto!

# A verdade...



Não  
histo  
essa  
ftware  
sto!

**Basta de reclamar  
que  
nada funciona!**

# Juntem-se a mim...

ultrad.com.br

**Em uma guerra  
contra os  
bugs de  
software!**



“That’s all folks!”