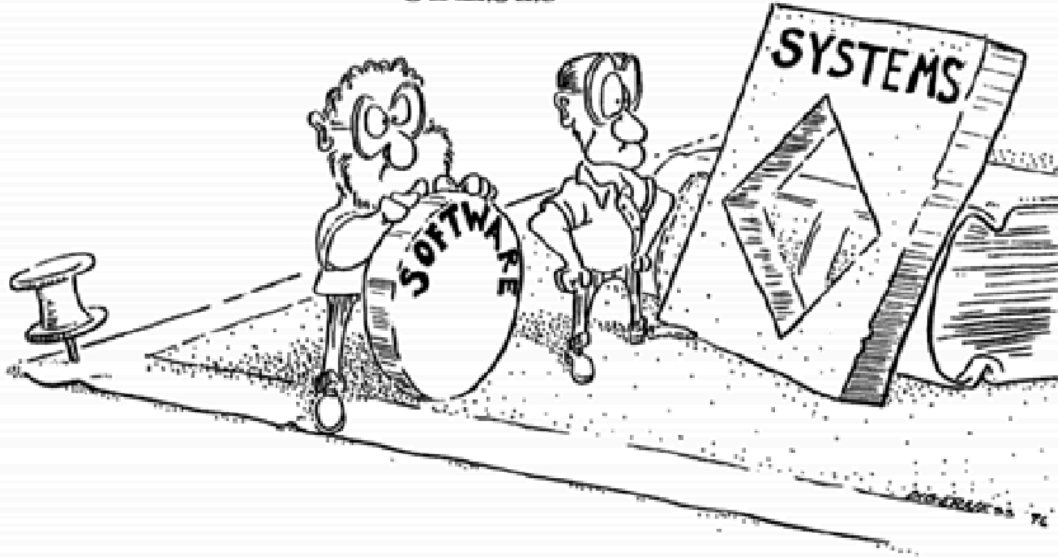


# TÓPICO 01: INTRODUÇÃO AO CÓDIGO LIMPO

Clean Code - Professor Ramon Venson - SATC 2026.2

## THE SOFTWARE CRISIS



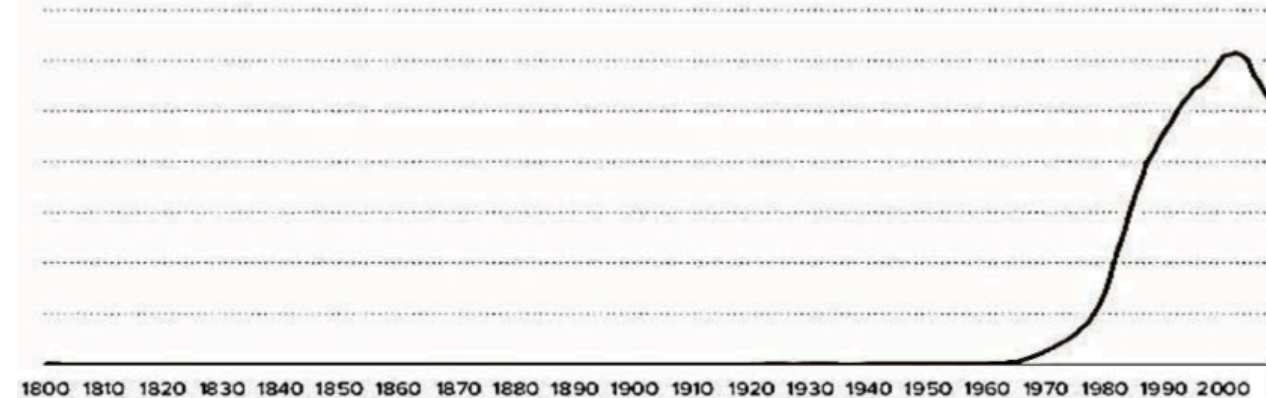
## A crise do software

Em 1968, a OTAN (Organização do Tratado do Atlântico Norte) criou um projeto para discutir e implementar padrões de desenvolvimento e especificar, pela primeira vez, o termo Engenharia de Software

## Aumento da demanda e da complexidade

Demanda crescente de novos softwares, impulsionada pelo estabelecimento de novas redes de computadores

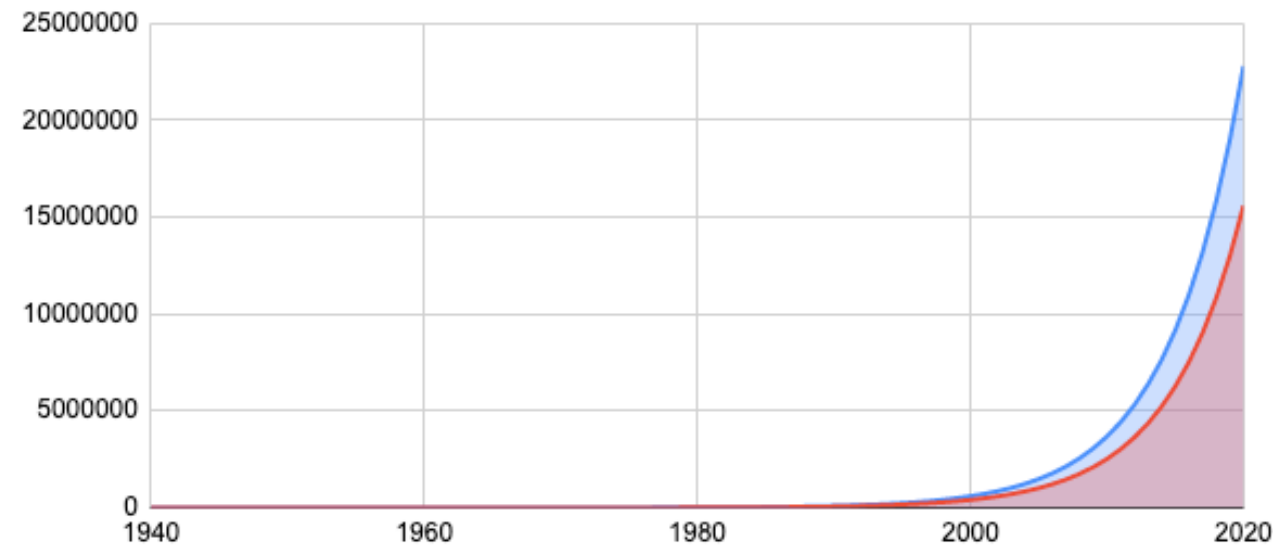
Frequency of 'Software' in Google Books



Total programmers and Software Developers



■ Total programmers ■ Software Developers



## Mesmos métodos e força de trabalho

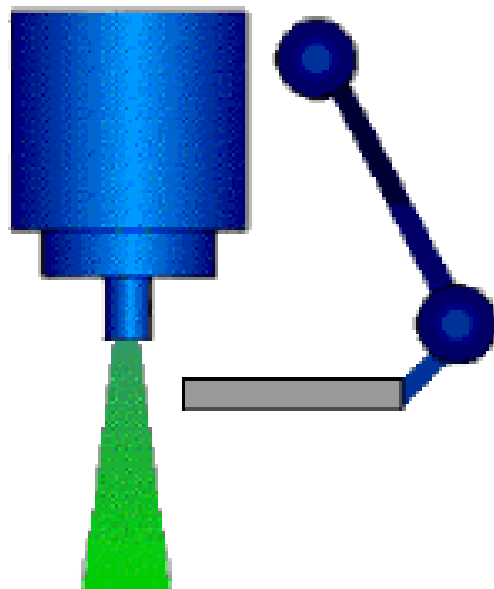
Necessidade de criar novos métodos e ferramentas de desenvolvimento que suportassem a demanda

## Caso Therac-25

Na década de 80, a Atomic Energy of Canada Limited (AECL) desenvolveu uma série de máquinas de radioterapia chamadas Therac

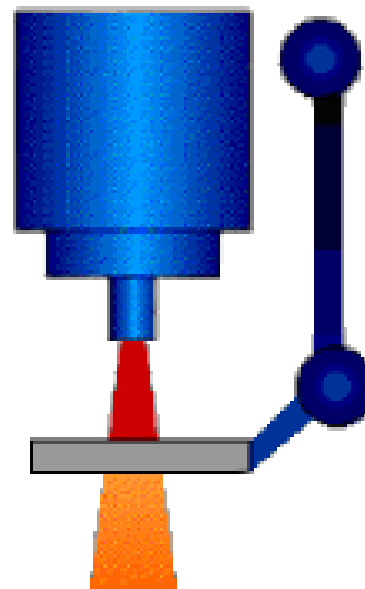


low current  
electron beam  
was scanned  
across the field



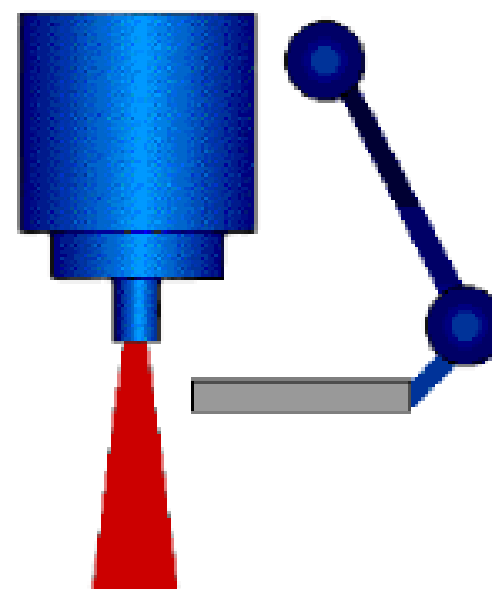
**Electron Mode**

high current  
electron beam  
was tracked  
at the target



**X-Ray Mode**

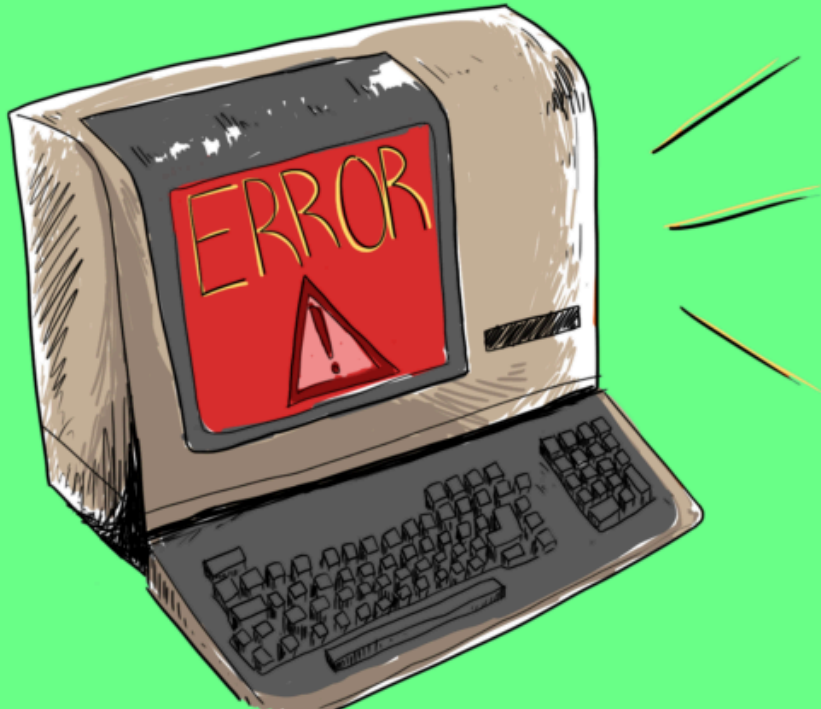
high current  
electron beam  
with no target  
> 'lightning'



**THE PROBLEM**

tray including the target, a flattening filter, the collimator jaws and an ion chamber was moved OUT for "electron" mode, and IN for "photon" mode.

6 mortes entre 1985 e 1987



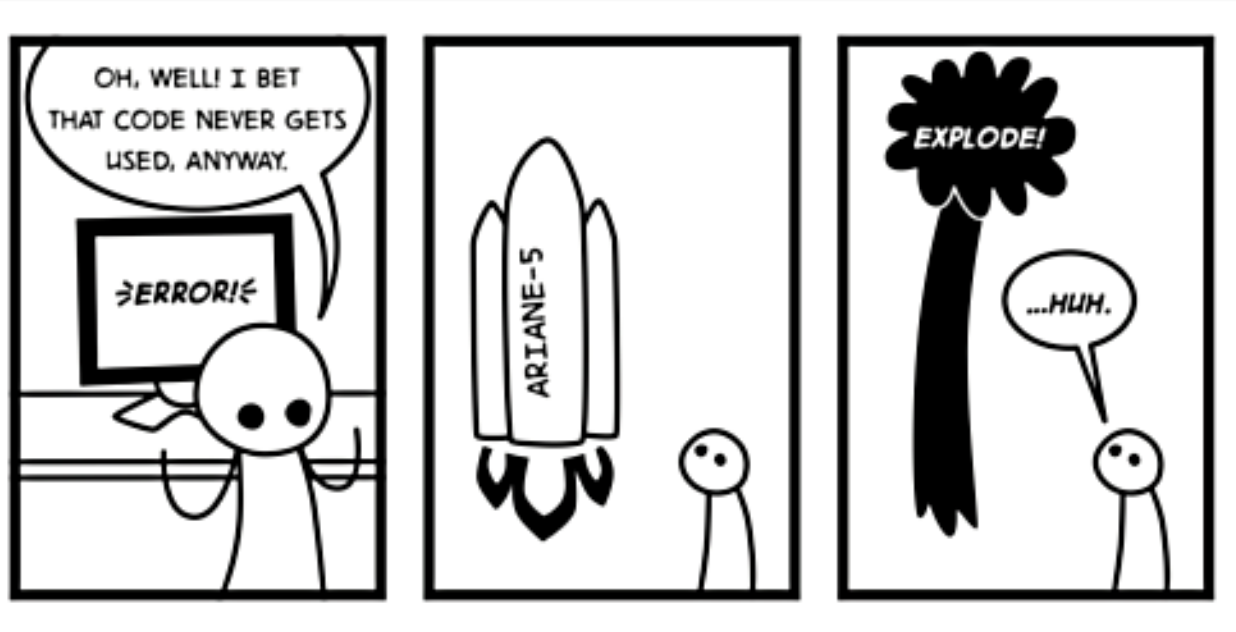
## Falhas de Software

- Design ruim
- Sem testes
- Sem mensagens de erro
- Excesso de confiança

## Caso Ariane 5

Em 1996, o foguete **Ariane 5** explodiu após 39 segundos de voo e causou um **prejuízo de 370 milhões de dólares**.





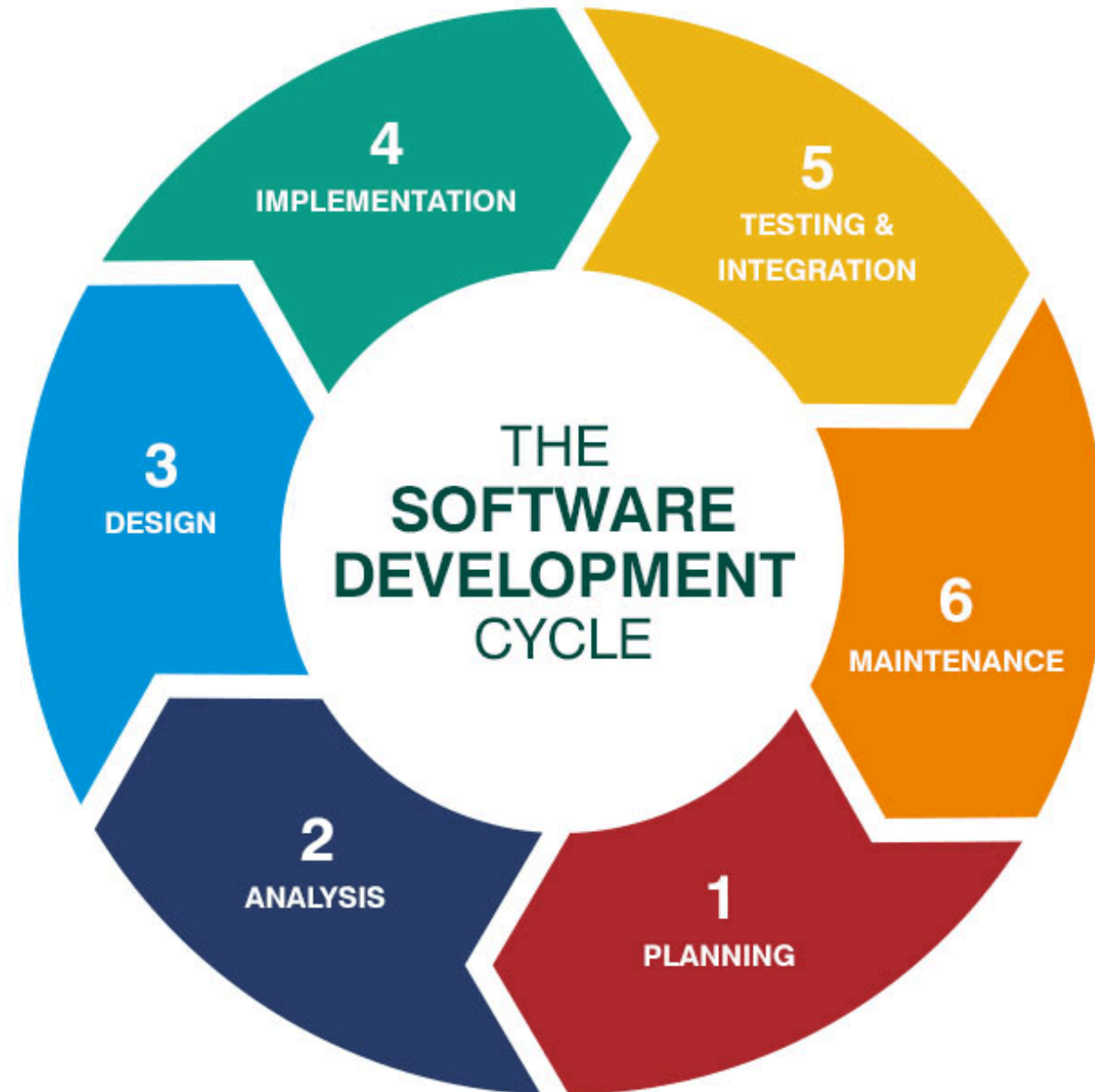
## Causa: Overflow

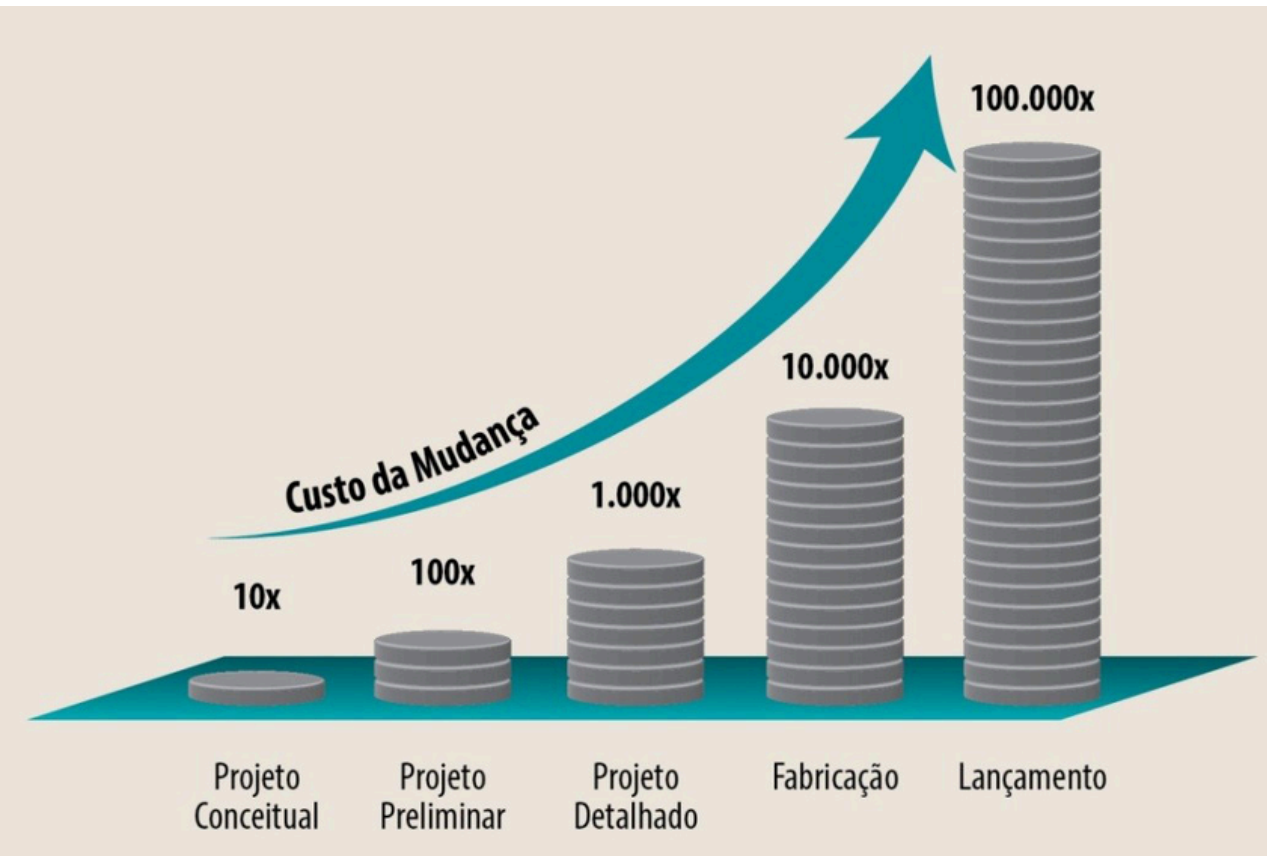
O sistema do foguete tentava converter uma variável de 64 bits em uma de 16 bits, resultando em um overflow do resultado em tempo de execução.

# Que outras falhas poderiam ser evitadas?

## Como evitar falhas?

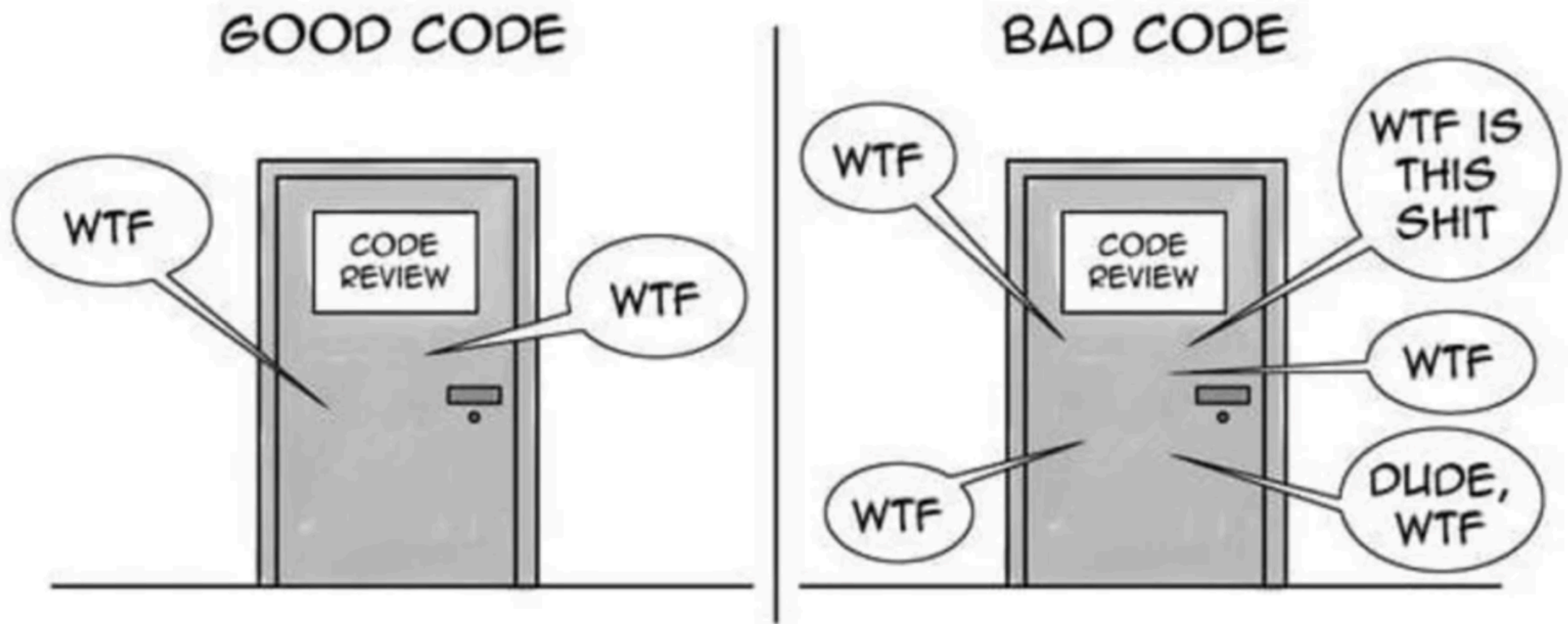
- Desenvolvimento Ágil
- Testes Automatizados
- Revisão de Código
- Arquitetura Limpa
- Programação Defensiva





## Impactos do Código Limpo

- Redução de custos
- Maior produtividade
- Menos vulnerabilidades
- Mais facilidade na manutenção
- Escalabilidade de equipes



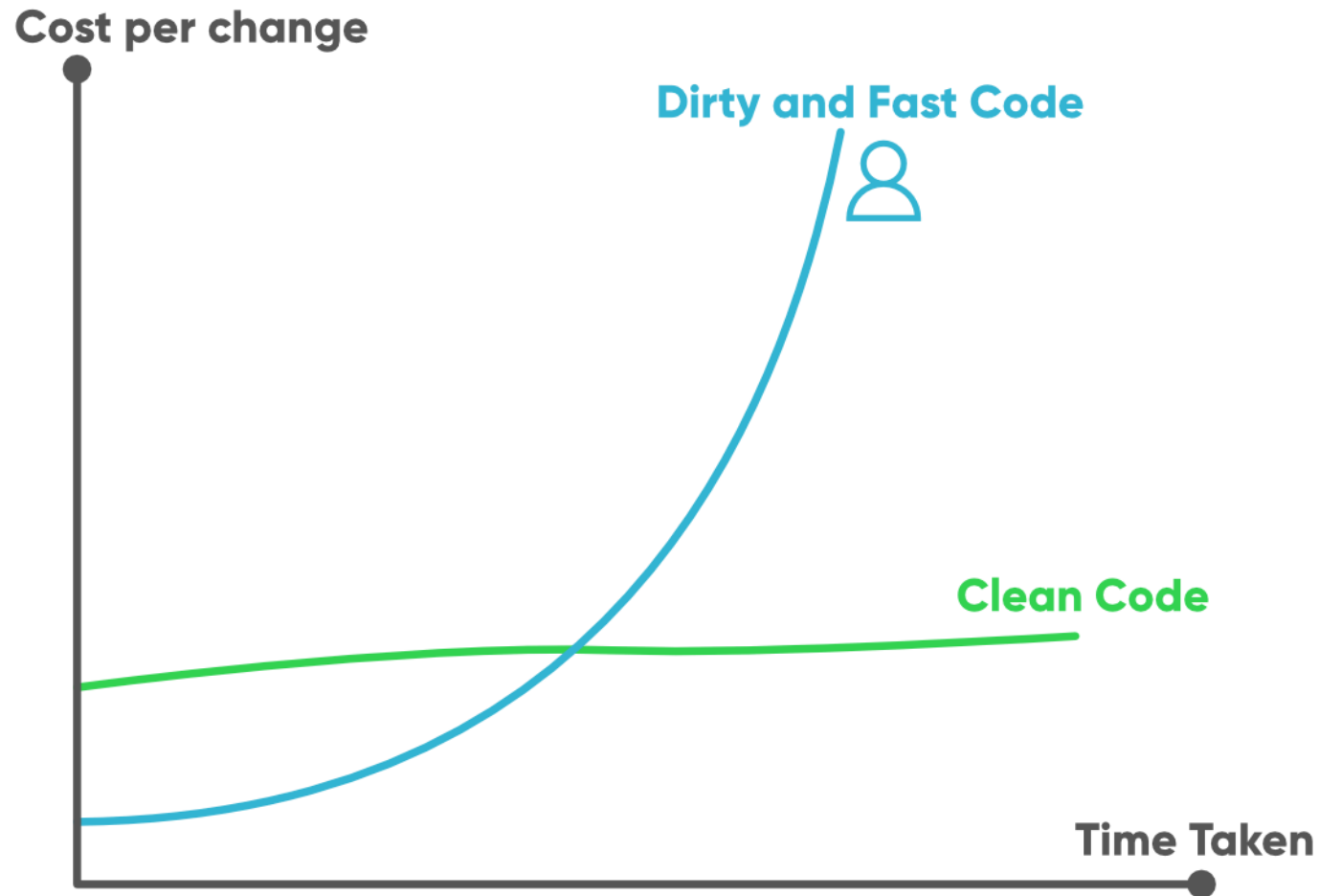
THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

# Código Limpo

O que faz um código ser considerado limpo?

- Legível (Readability)
- Mantível (Maintainability)
- Testável (Testability)
- Extensível (Extensibility)
- Elegante (Beauty)

```
$(window).scroll(function() {  
    let position = $(this).scrollTop() + that.headerHeight + 100;  
  
    if(position >= $('#content_7').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(6).addClass('active');  
    }else if(position >= $('#content_6').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(5).addClass('active');  
    }else if(position >= $('#content_5').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(4).addClass('active');  
    }else if(position >= $('#content_4').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(3).addClass('active');  
    }else if(position >= $('#content_3').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(2).addClass('active');  
    }else if(position >= $('#content_2').position().top){  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(1).addClass('active');  
    }else{  
        that.sidebarLi.removeClass('active');  
        that.sidebarLi.eq(0).addClass('active');  
    }  
});
```



## Barreiras do Código Limpo

O que faz algum programador não escrever código limpo?

- Ignorância
- Teimosia
- Arrogância
- Falta de Tempo

# É sua responsabilidade como programador defender seu código

Robert C. Martin

## Débito Técnico

O Débito Técnico é o resultado de uma implementação sem planejamento, mas que é adicionado para atender a uma demanda imediata.

Com o tempo, o débito técnico se acumula e pode levar a problemas de performance, segurança e escalabilidade.





## Analogia do Débito Bancário

O Débito Técnico é como um empréstimo bancário. Você pode pegar um empréstimo para comprar algo que precisa agora, mas terá que pagar juros no futuro.

O débito técnico as vezes compensa?

## Clean Code e LLMs

Sem o devido contexto, LLMs não conseguem inferir a **intenção** do código. Por isso, é importante escrever código limpo para que as LLMs possam entender e ajudar a melhorar o código.

Considere:

```
double calc(double a, double b)
```

## Melhorando o contexto para LLMs

```
double calculateFinalPrice(double productPrice, double taxRate)
```

A segunda versão dá ao modelo:

- Contexto de domínio
- Intenção do código
- Significado dos parâmetros

# Estudos científicos sobre o impacto do código limpo em LLMs

## Clean Code, Better Models: Enhancing LLM Performance with Smell-Cleaned Dataset

- Qualidade do código de entrada (com *code smells*) afeta fortemente o desempenho dos LLMs.
- Criação da ferramenta **SmellCC** (baseada em LLM) para remover *code smells* e validar em 50 repositórios.
- Resultados: **91,6%** dos *code smells* removidos em todo o CodeSearchNet-Python, **96,8%** de remoção com **91,3%** de correção em 50 repositórios, e ganhos de até **12,2%** em *code completion* e **4,3%** em *code search* com dados limpos.

# The Hidden Costs of Coding With Generative AI

Edward Anderson, Geoffrey Parker, and Burcu Tan

Generative AI can boost coding productivity, but careless deployment creates technical debt that cripples scalability and destabilizes systems.

## The Hidden Costs of Coding With Generative AI

- Modelos não reconhecem contexto implícitos
- Também acumula débito técnico
- Desenvolvedores júnior não possuem experiência para avaliar a qualidade do código gerado
- Necessita supervisão humana e maturação

)

## Does Code Cleanliness Affect Code Agents? A Controlled Minimal-Pair Study

### DOES CODE CLEANLINESS AFFECT CODING AGENTS? A CONTROLLED MINIMAL-PAIR STUDY

Priyansh Trivedi  
SonarSource  
priyansh.trivedi@sonarsource.com

Olivier Schmitt  
SonarSource  
olivier.schmitt@sonarsource.com

#### ABSTRACT

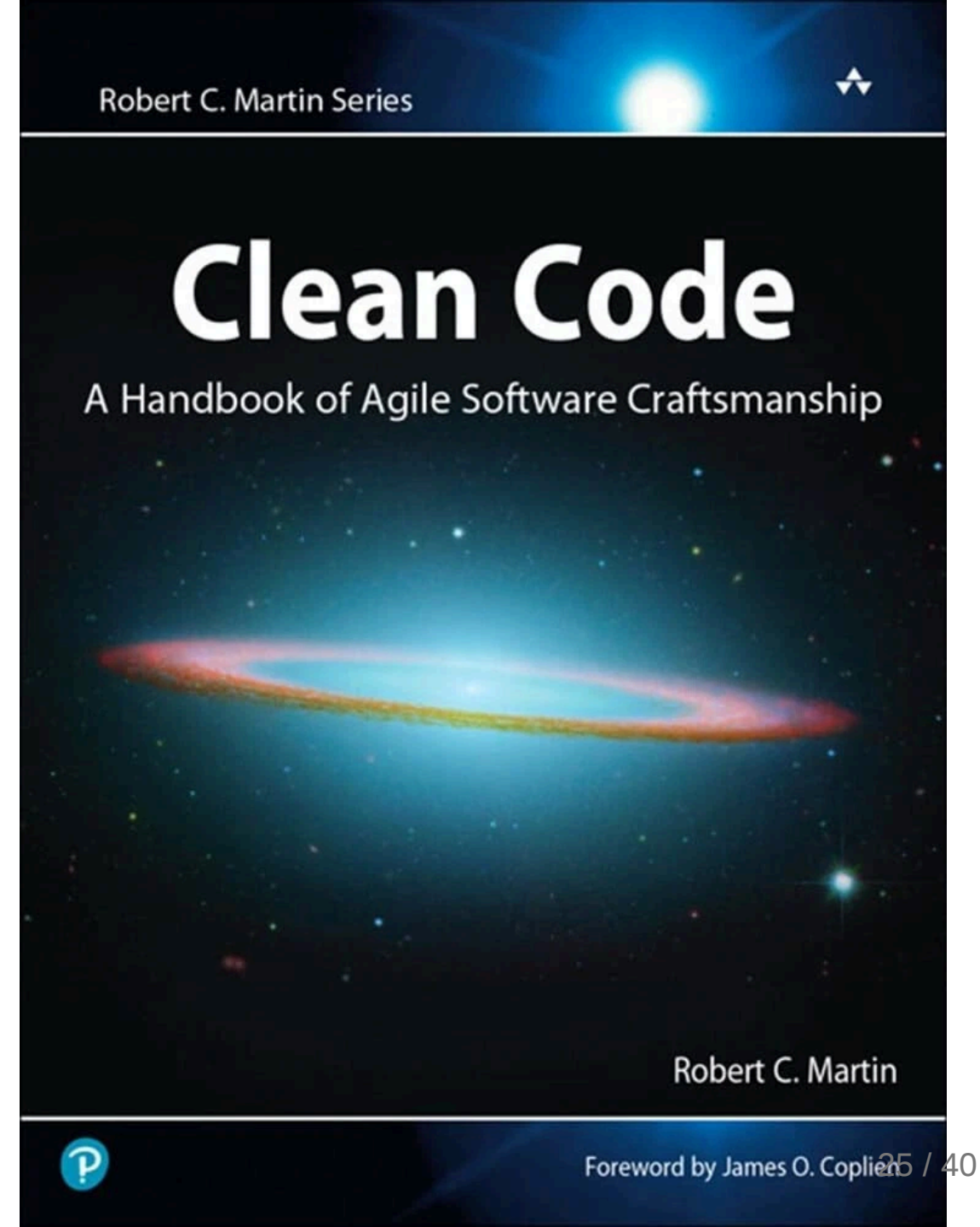
As autonomous coding agents see rapid adoption, their evaluation has primarily focused on task completion rates holding the target codebase fixed. This leaves a critical question unanswered: does the structural and stylistic quality, or “*cleanliness*” of the underlying code affect an agent’s ability to navigate and modify it? To isolate the effect of code cleanliness from agent capability, we introduce an evaluation protocol built around *minimal pairs*: repositories that match on architecture, dependencies, and external behaviour, but differ on static-analysis rule violations and cognitive complexity. The pairs are constructed in both directions, by agent pipelines that either degrade a clean repository or clean a messy one. We author 33 tasks across six such pairs, evaluated through hidden tests at the application’s public surface. Across 660 trials with Claude Code, code cleanliness does not change the agent’s pass rate. However, it substantially alters the agent’s operational footprint: agents working on cleaner code use 7 to 8% fewer tokens and reduce file revisitations by 34%. Our findings suggest that traditional maintainability principles remain highly relevant in the era of AI-driven development, shaping the computational cost and navigational efficiency of coding agents. Code cleanliness joins model choice, harness, and prompting as a factor that materially affects agent behaviours.

- Avaliou o impacto do código limpo em agentes de código (LLMs)
- Resultados: código limpo diminuiu o uso de tokens em 7-8% e diminuiu a revisão de arquivos em até 34%

# Práticas de Código Limpo

Livro escrito por Robert C. Martin  
cunhou o termo **Código Limpo**.

*É uma referência e não um dogma.*



| Name   | Label                |
|--------|----------------------|
| e3     | Have you bought a... |
| m_e3_0 | No vehicles          |
| m_e3_1 | Bicycle              |
| m_e3_2 | Motorbike            |
| m_e3_3 | Tuk-tuk?             |
| m_e3_4 | Other vehicle?       |

## Nomenclaturas

Existem apenas duas coisas difíceis em ciências da computação: Invalidação de cache e dar nome as coisas - Phil Karlton, Netscape developer

```
9 create or replace table COMPANY_NAME.somedata AS
8 -- ...
7 create or replace table COMPANY_NAME.moredata AS
6 -- ...
5 create or replace table COMPANY_NAME.moremoredata AS
4 -- ...
3 create or replace table COMPANY_NAME.datadata AS
2 -- ...
1 create or replace table COMPANY_NAME.somemoredata AS
10 -- ...
```

```
private int  
{  
    try  
    {  
        bool ehLoja = new  
  
        int lojaId = 0;  
  
        if (!ehLoja)  
        {
```



**Descodando | Programação e Tec...** · 21m ...

Não sei vocês, mas eu tenho costume de dar nome de pessoas para as variáveis

```
1  ✓ def soma(carlos, maria):  
2      ana = carlos + maria  
3      return ana  
4  
5  joaquim = soma(5, 7)  
6  
7  print("A soma é:", joaquim)
```



```
// Case 1: Double numbers in an array
function doubleArray(array) {
  return array.map(number => number * 2)
}

// Case 2: Simple subtraction
function subtract(x, y) {
  return x - y;
}

// Case 3: Simple multiplication
function multiply(x, y) {
  return x * y;
}
```

## Funções e Comentários

Uma função bem escrita vale mais do que mil comentários. Um comentário pode mal escrito pode valer por uma função mal escrita.

```
0 references
public bool IsEven(int foo) {
    if (foo < 0) {
        foo *= -1;
    }
    for(int i = 0; i <=foo; i += 2)
    {
        if (i == foo) {
            return true;
        }
    }
    for (int i = 1; i <= foo; i += 2)
    {
        if (i == foo)
        {
            return false;
        }
    }
    return (foo % 2 == 0);
}
```

```
def date_warning(): #warn students that there book is not yet returned
    #for a day or two or more
    borrow_records = []
    borrow_records.append(get_borrow_data()) #Appending the loaded json to to be incremented
    for x in borrow_records: #First increment
        for b in x: #Second increment Note: Should have use the json dumps or json loads
            current_datetime = datetime.now() #Get the current time and date
            ret_date = b['date_returned'] #return date
            ret_time = b['time_returned'] #return time

            return_stat = b['return_status'] #return status boolean true or false
            #return_stat is only true if a book is returned and false if not

            date_time_ret = f'{ret_date} {ret_time}' #Combine both into a string

            #turn date_time_ret into a strptime formats
            initial_ret = datetime.strptime(date_time_ret, "%Y/%m/%d %I:%M:%p")
            current_datetime = datetime.now() #Get current time and date

            #Calculate the total amount of hours to be calculated and turned into fines
            current_data = (current_datetime - initial_ret).total_seconds() / 3600
            if current_data != 0 and return_stat == False: #Sending a message if the return_stat = false
                #And the current_data !=0 means that if its 0 hence it still has time left to be returned
                print("Please return the book")
```

## Formatação e Estrutura

Grandes empresas de software possuem padrões de formatação de código.

- [Google Java Style Guide](#)

```
,a:active{outline:0}abbr[title]{  
bold}blockquote{margin:1em 40px}d  
or:#000}pre,code,kbd,samp{font-fa  
r new', monospace;font-size:1em}p  
-wrap:break-word}q{quotes:none}q:  
font-size:75%}sub,sup{font-size:7  
lign:baseline}sup{top:-0.5em}sub{  
in:0 0 0 40px}nav ul,nav ol{list-  
bicubic}svg:not(:root){overflow:h  
argin:0 2px;padding:0.35em 0.625e  
tton,input,select,textarea{font-s  
tical-align:middle}button,input{l  
n,table input{*overflow:auto}butt  
l,input[type="submit"]{cursor:poi  
checkbox"},input[type="radio"]{bo  
kit-appearance:textfield;-moz-box  
box;box-sizing:content-box}input[  
webkit-appearance:none}button::-m  
0;padding:0}textarea{overflow:aut  
lapse;border-spacing:0}@media scr
```

19 group by 1;

Expected indent of 4 spaces. (LT02)

[View Problem \(⌘F8\)](#) No quick fixes available

```
23 select generate_series(  
24     date_trunc('hour', now()) - '1 day'::interval,  
25     date_trunc('hour', now()),  
26     '1 hour'::interval  
27 ) as hour  
28 )
```

30 select

92

9/9

0800 Anttan started

1000 " stopped - anttan ✓

1300 (032) MP-MC { 1.2700 · 9.037 847 025  
~~1.982647000~~ 9.037 846 895 correct  
~~2.130476415~~ 4.615925059(-2)

(033) PRO 2 2.130476415  
 correct 2.130676415

Relays 6-2 in 033 failed special speed test  
 in Relay " 10.000 test.

Relays changed

1700 Started Cosine Tape (Sine check)

1525 Started Multi-Adder Test.

1545  Relay #70 Panel F  
 (moth) in relay.

First actual case of bug being found.

1630 Anttan started.

1700 closed down.

Relay 3145  
 Relay 3370

## Tratamento de Erros

O primeiro bug de software registrado foi, literalmente, um inseto preso em um computador em 1947, documentado por Grace Hopper.



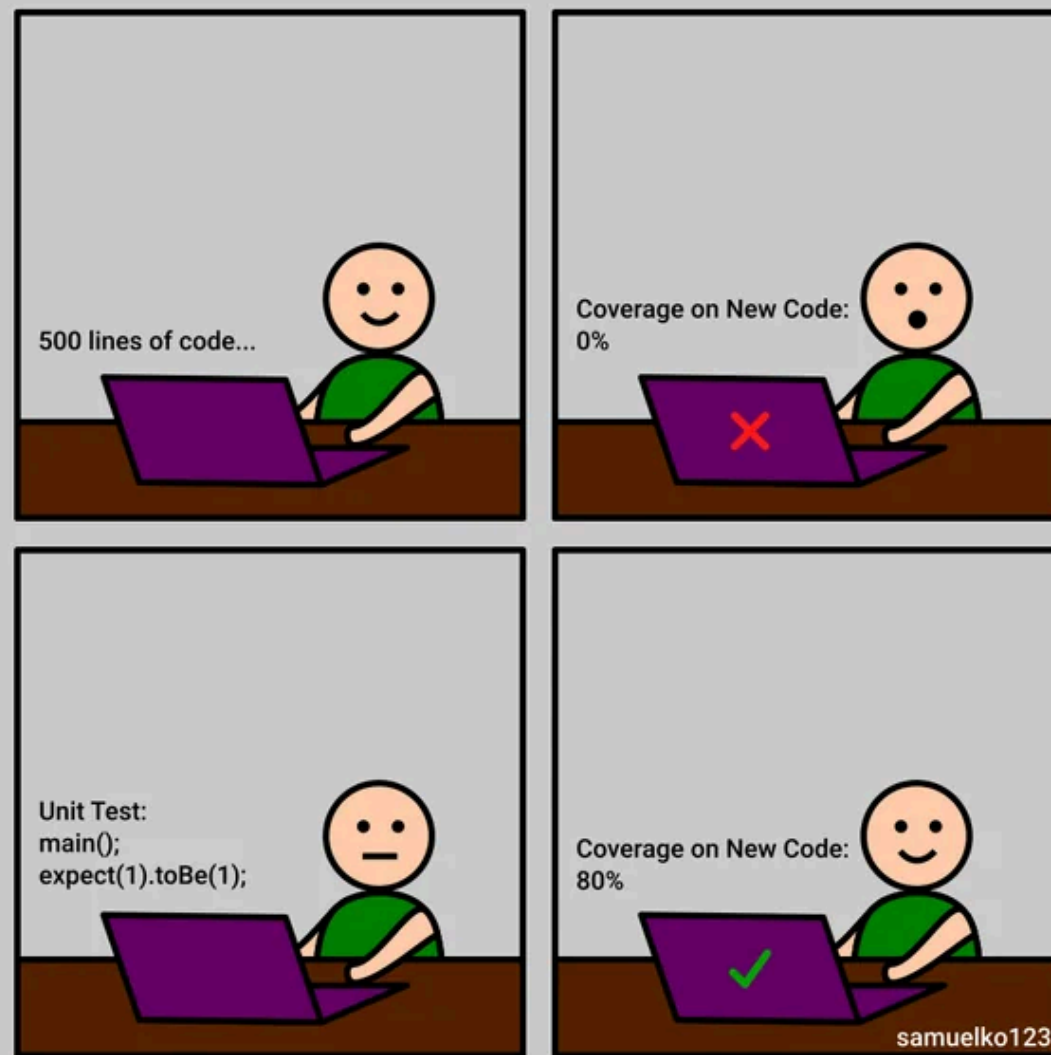
```
public static void main(String[] args) {  
    try {  
        InputStream file = new FileInputStream(new File(args[0]));  
    }  
    catch (FileNotFoundException e){  
        System.out.println("Could not locate file, please try again");  
    }  
}
```

```
public static void main(String[] args) throws Exception {  
    InputStream file = new FileInputStream(new File(args[0]));  
}
```

## Estruturas de Dados

Qual a melhor estrutura de dados para abstração de um problema?





Escrever código limpo é uma arte,  
não uma ciência.

## Material de Apoio

- Clean Code (PT/EN)
- Clean Code, Better Models: Enhancing LLM Performance with Smell-Cleaned Dataset
- As falhas numéricas que podem causar desastres
- The Hidden Costs of Coding With Generative AI
- Clean Code, Better Models: Enhancing LLM Performance with Smell-Cleaned Dataset