



# Teste de Software

**UNIP - Araraquara**

**Curso:** Ciências da Computação

**Disciplina:** Engenharia de Software

**Profº:** João Paulo Moreira dos Santos

10/10/2  
017

# Desastres causados por erros

- Em 1996 - Um software com uma exceção não tratada foi responsável pela explosão do **foguete Ariane-5**, quando a 40 seg após a iniciação da sequência de voo, o foguete se desviou de sua rota, partiu e explodiu.



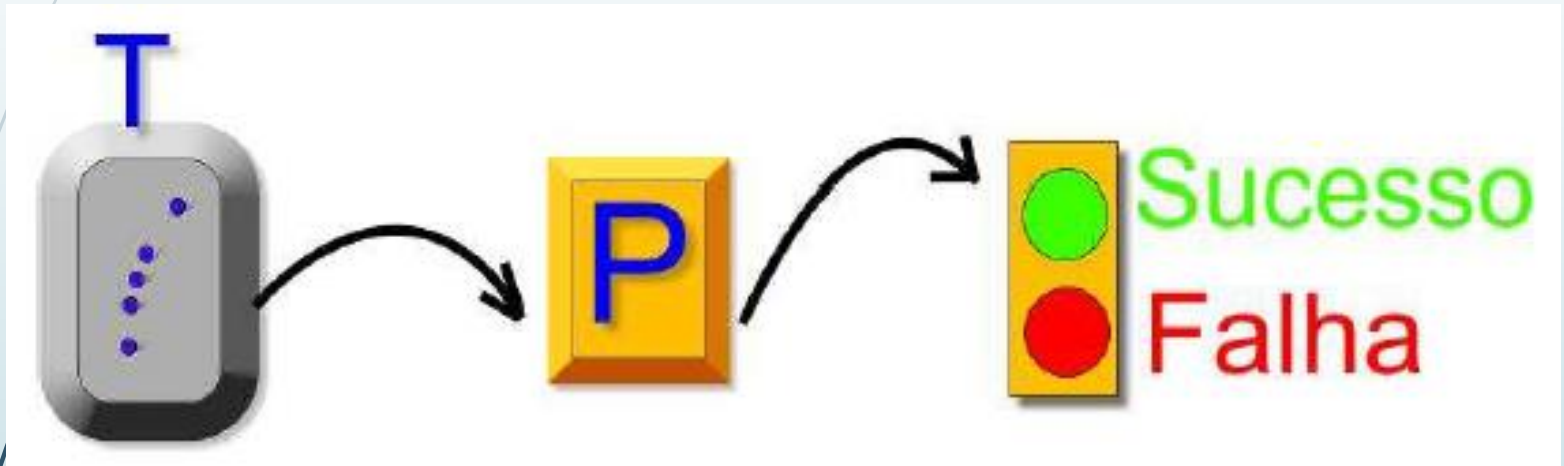
# Desastres causados por erros

- Em 2000: Erro de cálculo no **sistema de radioterapia**, que era utilizado para controlar a emissão de radiação em tratamentos de câncer **matou 8 pessoas e causou queimaduras graves em outras 20.**



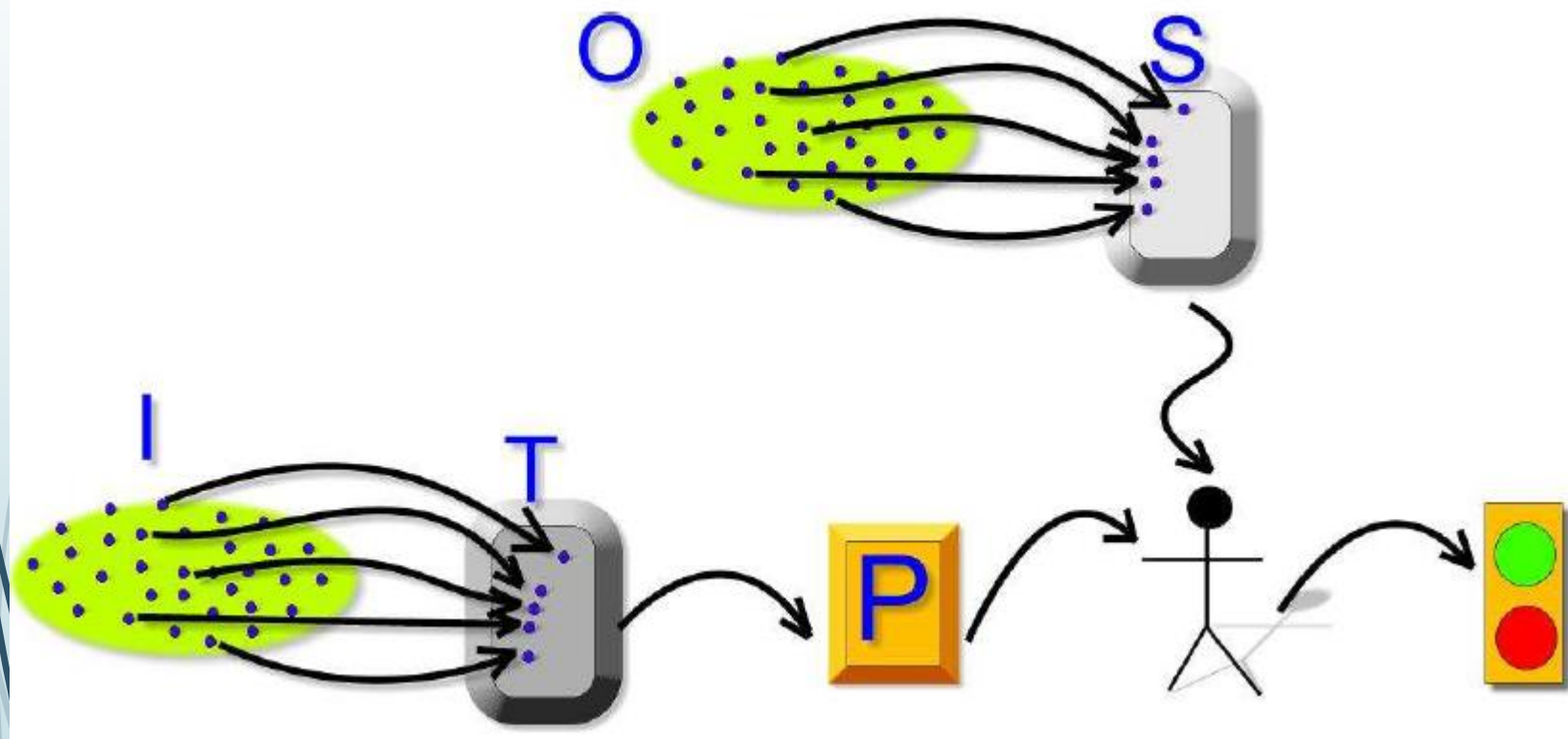
# O que é Teste de Software ?

- Atividade de executar um programa e verificar se o seu comportamento é o esperado.



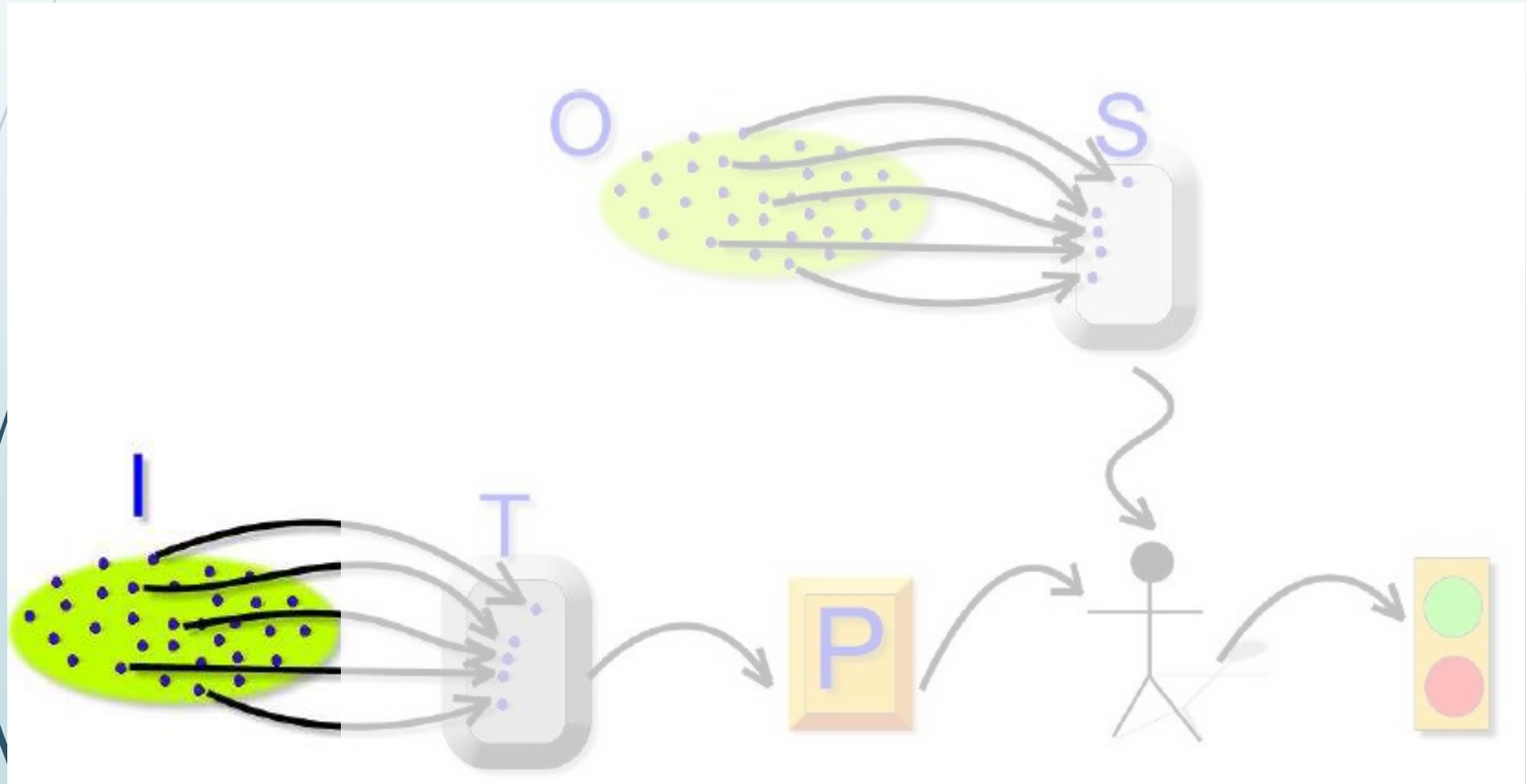
- Objetivo: Revelar defeitos

# O que é Teste de Software ?



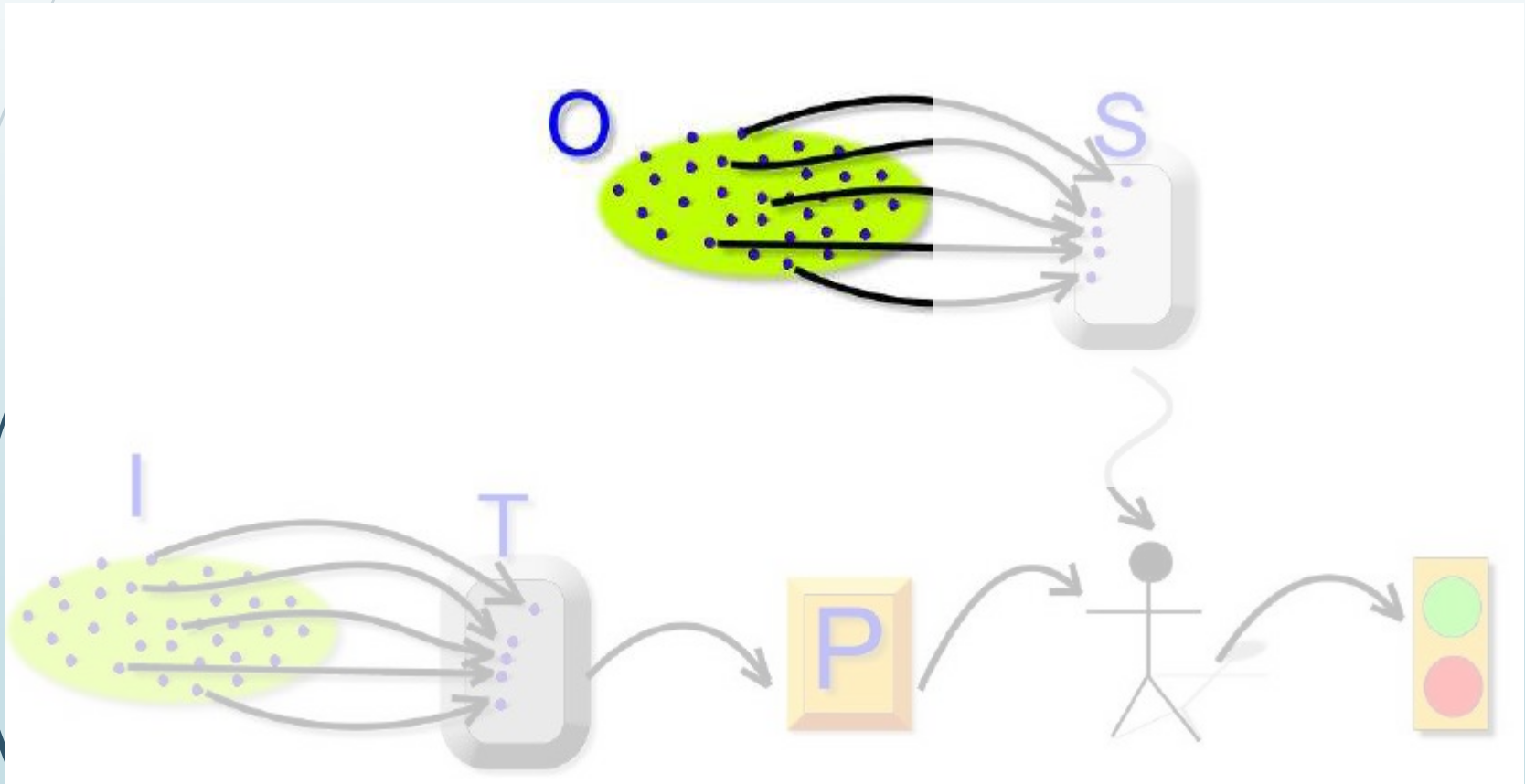
# O que é Teste de Software ?

- **Domínio de Entrada:** conjunto de todos os dados que o programa deve tratar.



# O que é Teste de Software ?

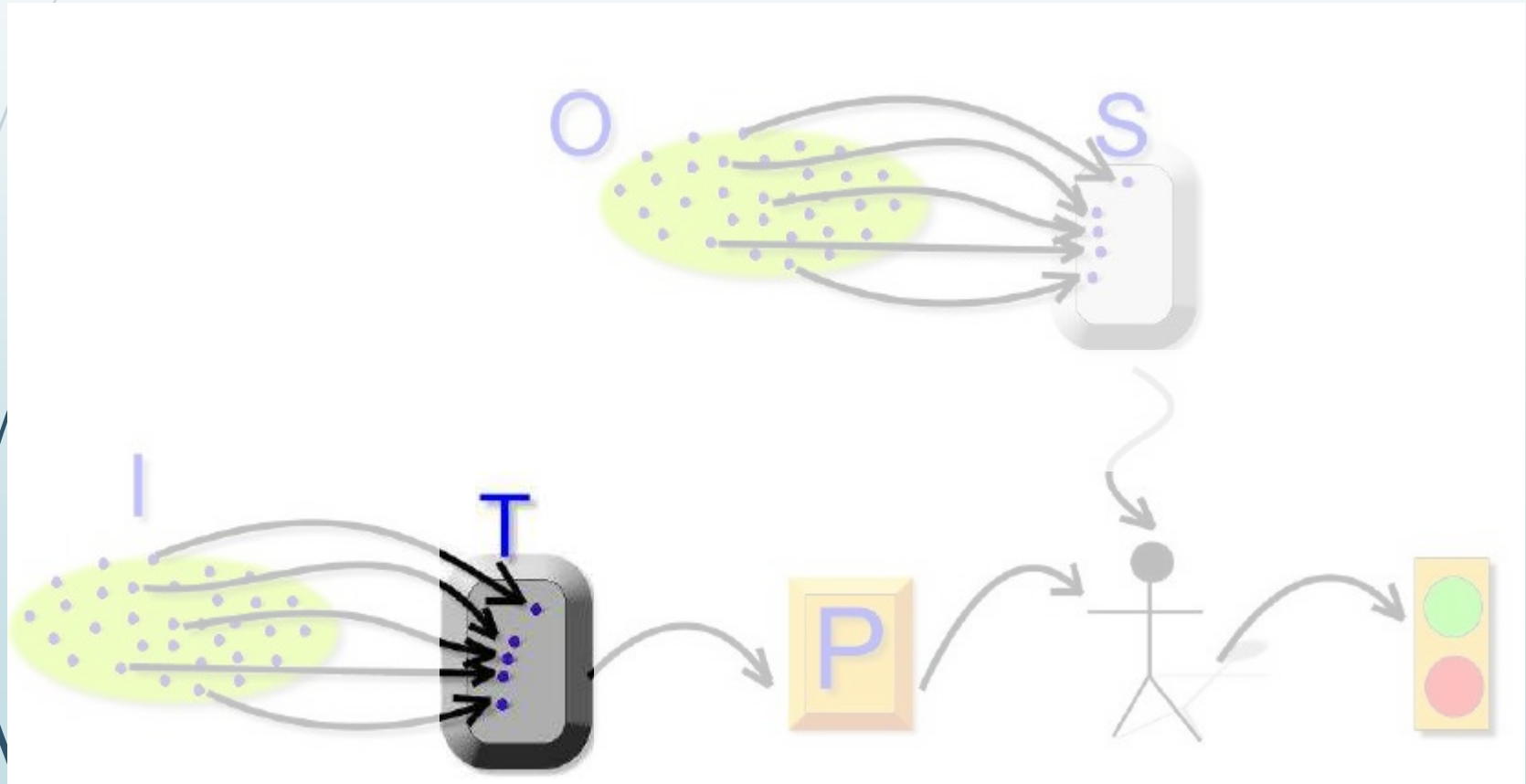
- **Domínio de Saída:** todos os possíveis resultados que o programa deve fornecer.





# O que é Teste de Software ?

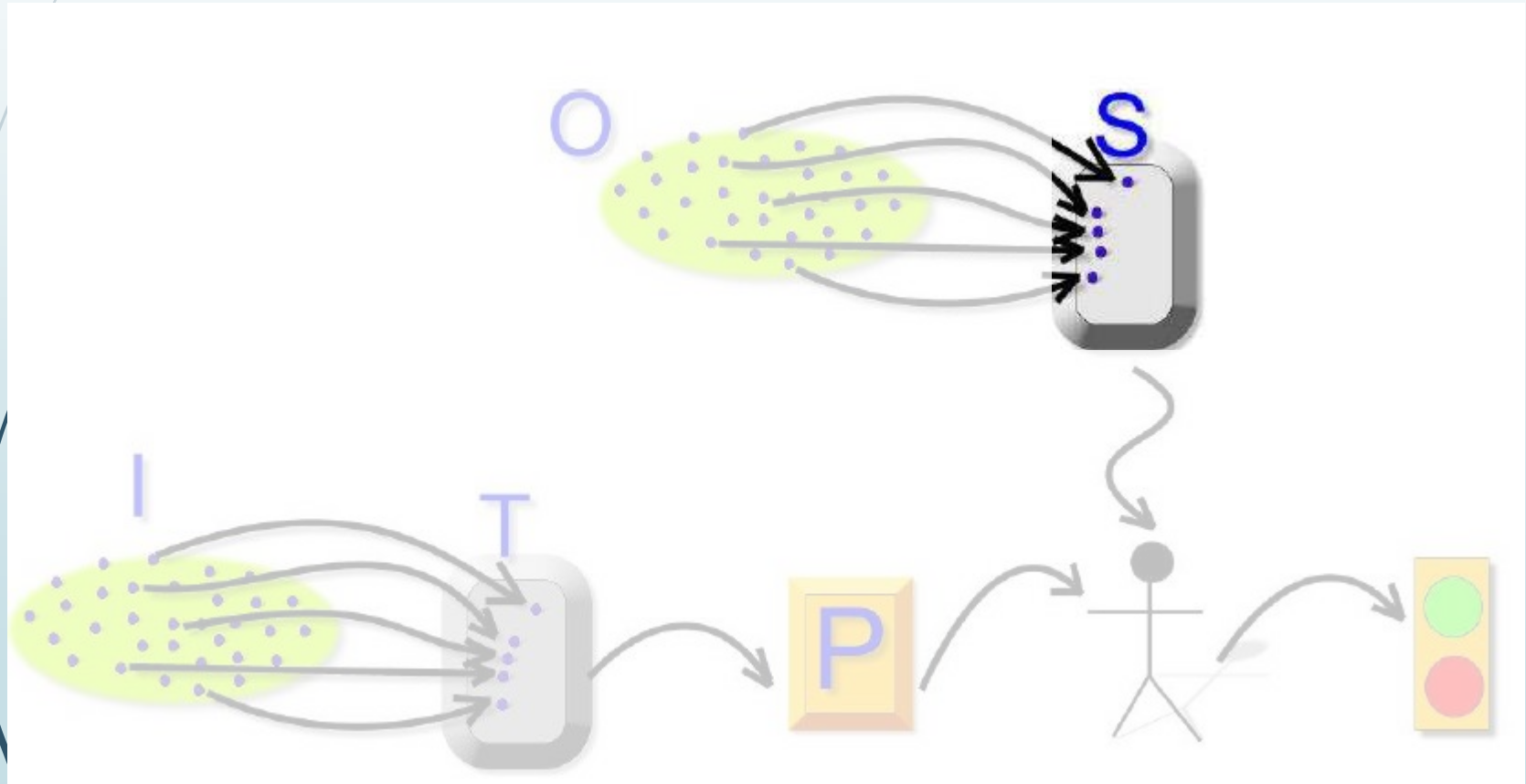
- **Dados de Teste:** subconjunto do domínio de entrada.





# O que é Teste de Software ?

- **Resultados esperados:** subconjunto correspondente do domínio de saída.



# O que é Teste de Software ?

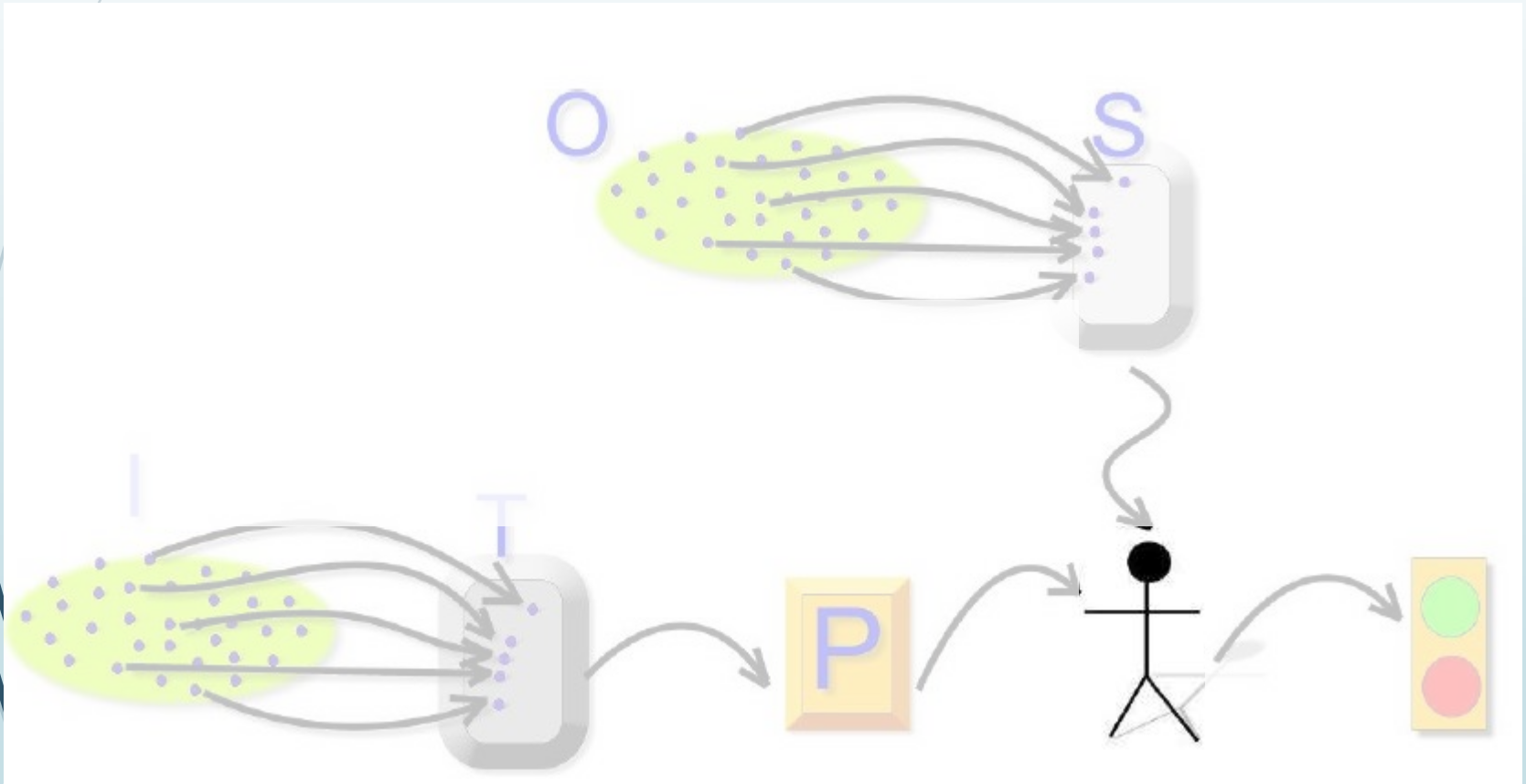
- **Casos de teste:** é um par composto pelo dado de entrada e pelo resultado esperado.
- Ex:
  - *DT1: 5, 4, 3*
  - *CT1: { 5, 4, 3 (Escaleno) }*

```
a = input('Digite o tamanho do primeiro lado: ')
b = input('Digite o segundo lado: ')
c = input('Digite o terceiro lado: ')

if a + b > c:
    if a == b and a == c:
        print 'Triângulo Equilátero'
    elif a == b or b == c or a == c:
        print 'Triângulo Isósceles'
    elif a != b and c or b != a and c or a != c:
        print 'Triângulo Escaleno'
else:
    print 'É impossível ser um triângulo'
```

# O que é Teste de Software ?

- **Oráculo:** mecanismo que decide sobre a correção de uma execução.



# Por que testar?

- Para assegurar que as necessidades dos usuários estejam sendo atendidas.
- Porque é provável que o software possua defeitos.
- Desenvolvedor já alocado para outro projeto teria que resolver muitos bugs de projetos anteriores em produção.
- Porque falhas podem custar muito caro.
- Para avaliar a qualidade do software.

# Erro, Defeito e Falha

- **Erro (*error*):** é fruto da ação humana, que produz um resultado incorreto.
- **Defeito (*fault*):** também conhecido como bug, é o resultado de um erro no código, gerando uma anomalia no funcionamento no sistema.
- **Falha (*failure*):** é resultado da execução de um defeito no código.

# Erro, Defeito e Falha



Uma  
pessoa  
comete um  
**erro**...

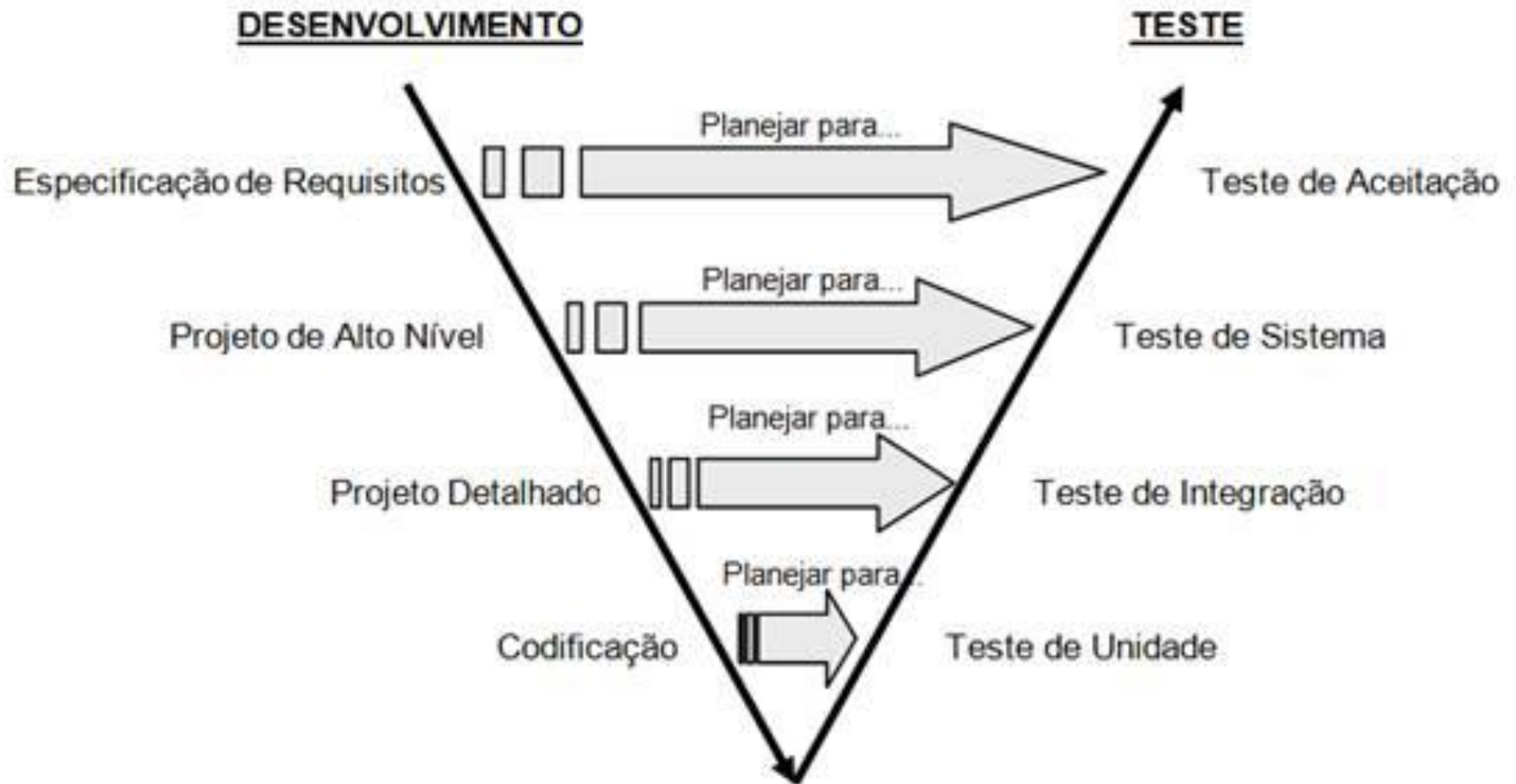


... que cria  
um **defeito**  
no  
software...



... que  
pode  
causar uma  
**falha** na  
operação

# Níveis do Teste





# Níveis do Teste - Teste de Aceitação

- Verifica se o sistema está em conformidade com os requisitos esperados pelo cliente.
- Realizado pelo **cliente** ou pelo **testador**.
  - **checklist** feito pelo cliente do que é esperado que haja no sistema.
- O sistema é utilizado para capacitação dos usuários de forma que eles validem todos os requisitos do sistema.
- Realizado de forma **manual** ou **automática**.

# Níveis do Teste - Teste de Sistema

- Verifica se o sistema está em conformidade com a especificação de requisitos;
- Realizado pelo **testador**, o qual tem acesso apenas a interface do sistema;
- Os testes geralmente são baseados em **roteiros de teste** criados a partir da **especificação**;
- Pode ser realizado de forma **manual** ou **automática**;
- Ferramentas: Selenium, Watir, Badboy.

# Níveis do Teste - Teste de Integração

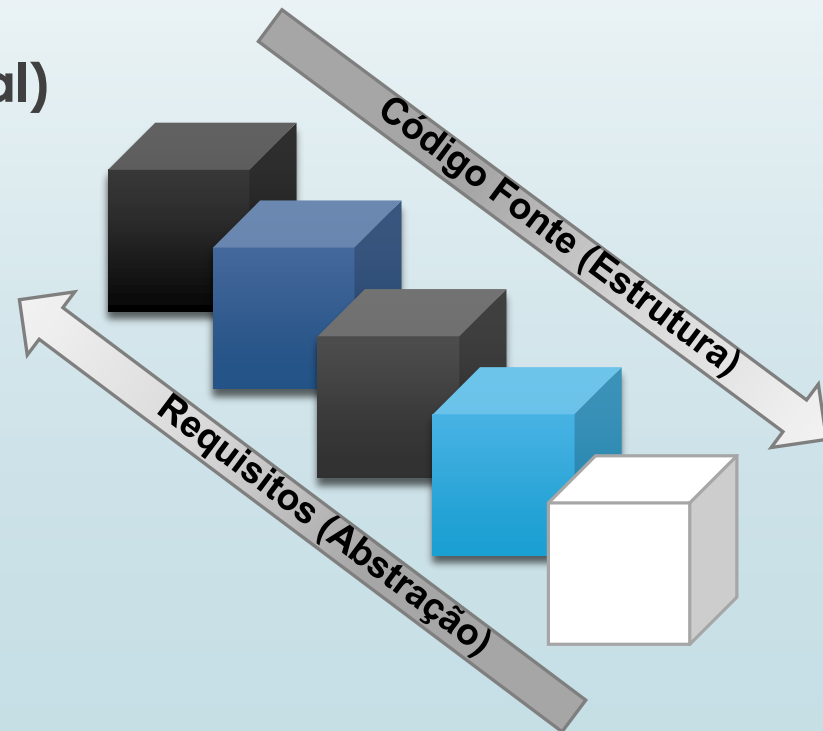
- Estratégia de integração incremental;
- Verifica se ao juntar vários componentes do sistema, se eles se comunicam corretamente;
- A interface entre as unidades é testada;
- Realizado pelos **desenvolvedores** ou analistas de sistema para testar um **módulo do sistema**;
- Realizado de forma **automática**.

# Níveis do Teste - Teste de Unidade

- Teste realizado em uma unidade ou componente para verificar sua corretude.
  - Ex.: Teste para uma classe ou método do sistema.
- Características avaliadas.
  - interface, compatibilidade, caminhos de execução, atendimento a erros, condições de contorno.
- Deve ser automático, completo, independente, reproduzível.
- Para Java, existe a ferramenta **Junit**.
- Realizado de forma **automática**.

# Técnicas do Teste

- Testes de Caixa-Branca (Estrutural)
  - Testes de Unidade
  - Teste de Integração
- Testes de Caixa-Preta (Funcional)
  - Testes Funcionais
  - Testes de Aceitação
  - Testes Exploratórios
- Baseado em Defeitos
  - Teste de Mutação



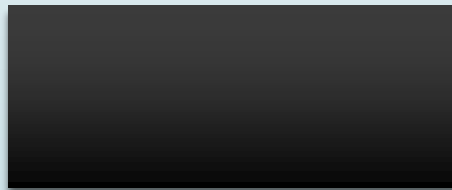
# Técnicas do Teste – Funcional

- Identificar as **funcionalidades** que o software deve realizar;
- Projetar **casos de teste** capazes de verificar se essas funcionalidades estão sendo realizadas pelo software;

Requisito	Descrição
RF01	Login do Administrador
RF02	Morador solicita entrada no condomínio
RF03	Cadastrar Moradores
RF04	Consultar dados dos Moradores registrados no Banco de Dados
RF05	Autenticar todos os dados do Morador
RF06	Login do Morador
RF07	Relatório de Dados
RF08	Permitir acesso para Moradores registrados no Banco de Dados
RF09	Liberar sinal para o sistema liberar portão

**Requisitos**

*Entrada*



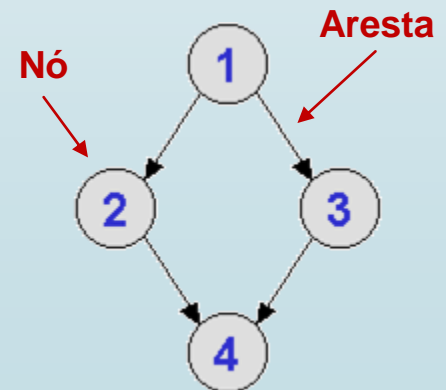
*Saída*



**Resultados Esperados**

# Técnicas do Teste – Estrutural

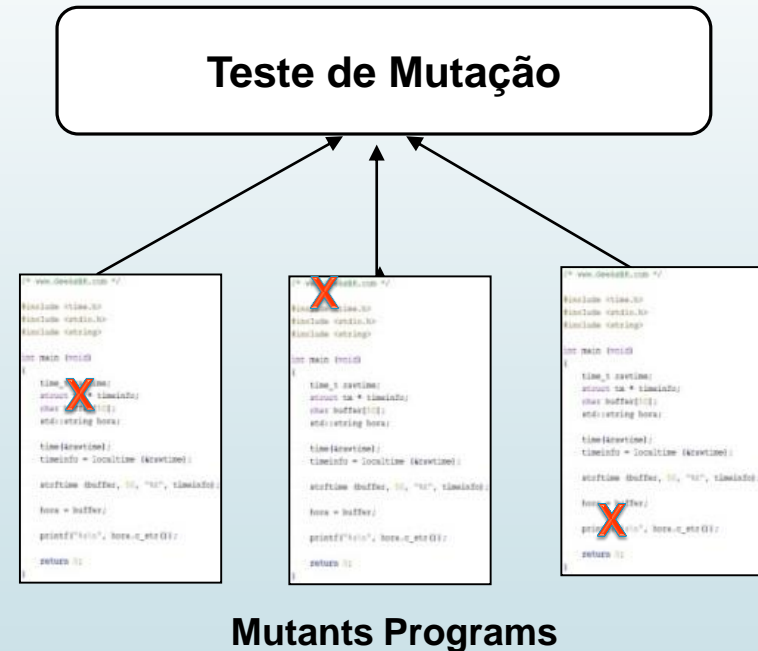
- **Requisitos de testes** com base no **código fonte**;
- Relacionada ao conhecimento da **estrutura interna do programa**;
- Utilizam uma representação de programa conhecida como **Grafo de Fluxo de Controle (GFC)**.





# Técnicas do Teste – Baseado em Defeitos

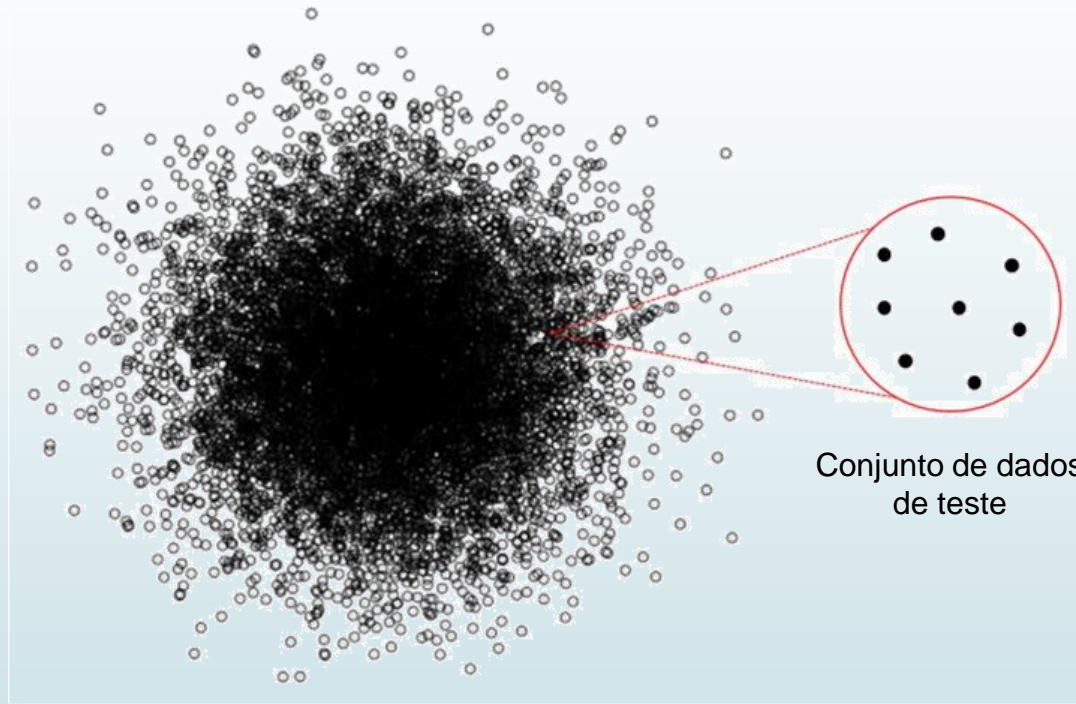
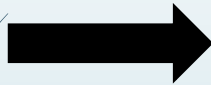
- Visa a medir a adequação de um conjunto de casos de teste;
- Utiliza um conjunto de programas com erros introduzidos em sua estrutura, criando diversas versões denominadas de **mutantes**;
- O objetivo é fazer que o programa mutante falhe, demonstrando a **eficácia dos casos de teste**;



# Geração de Dados de Teste

É um processo de identificação de **dados do domínio de entrada válidos** para um determinado programa de acordo com os **critérios de teste**.

# Geração de Dados de Teste – Processo



Domínio de entrada

# Geração de Dados de Teste – Por que automatizar?

► Na Prática....

## 1. Programador implementa a funcionalidade



# Geração de Dados de Teste – Por que automatizar?

► Na Prática....

**2. O testador inicia os testes das funcionalidades implementadas e ....**



# Geração de Dados de Teste – Por que automatizar?

► Na Prática....


## 3. Gera os dados e executa os testes **MANUALMENTE**





# Geração de Dados de Teste

Por que, então, **NÃO**  
automatizar essa atividade?





# Geração de Dados de Teste

Geração Manual	Geração Automática
Muitos dados de teste	Poucos dados de teste
Muito trabalhosa e complexa	Reduz significativamente o esforço
Demanda muito tempo	Mais rápida
Dispendiosa	O custo é reduzido
Sujeito a falhas	Alto grau de confiabilidade
	Detecção de regressões

# Geração de Dados de Teste Automatizada

- É fundamental para o teste de software, pois os **dados de teste** determinam a **qualidade do teste** durante sua execução;
- É uma tarefa **desejável**, porém **não trivial**, em função da **complexidade** de determinados domínios:
  - Tamanho do programa
  - Caminho ausente
  - Caminhos não executáveis
  - Mutantes equivalentes