INSTITUTO FEDERAL
Pernambuco
Campus
Jaboatão dos
Guararapes

# Testes de Software

Prof: Nilson Júnior
nilson.junior@jaboatao.ifpe.edu.br

www.professornilson.wix.com/ifpe

# Agenda

- Apresentar uma introdução sobre testes de softwares dentro do ciclo de vida de software.
- Abordar Técnicas usadas nas atividade de testes.
- Conhecer um processo de testes de software.

# O Problema

- O desenvolvimento de software utilizando as metodologias, técnicas e ferramentas da Engenharia de Software não oferece a total garantia de qualidade do produto obtido, apesar de melhorá-la significativamente.

# O que são os Testes de Software?

# Introdução

- Teste é um conjunto de atividades que pode ser planejado antecipadamente e realizado sistematicamente.

- É possível definir um “template” (esqueleto), ou seja um conjunto de passos ao qual é possível alocar técnicas de projeto de casos de teste e estratégias de teste específicos.

# Testes de Software

- Por esta razão é uma etapa fundamental na obtenção de um alto nível de qualidade do software a ser produzido é aquela onde são realizados os procedimentos de teste, uma vez que esta é a última etapa de revisão da especificação, do projeto e da codificação.

## Tópicos Importantes:

- Realização de forma cuidadosa e criteriosa, dos procedimentos associados ao teste de um software.

- Dar uma importância cada vez maior ao impacto que os testes de software causam ao funcionamento do software.

Por esta razão, o esforço despendido para realizar a etapa de teste pode chegar a 40% do esforço total empregado no desenvolvimento do software.

# Motivos que justificam os testes de software

- Quando um produto não é testado, há uma grande chance que ele contenha erros ou defeitos, assim este produto não vai satisfazer as necessidades do cliente;
- O cliente quando não está satisfeito com o produto, dificilmente irá contratar novamente a empresa;
- Quando não há qualidade no produto, a empresa fica com a imagem prejudicada. Logo, é necessário mais investimento em marketing, por exemplo;
- Se os erros são encontrados pelo cliente, o custo para corrigir esses erros é muito maior do que quando o sistema ainda está em desenvolvimento;

# Motivos que justificam os testes de software

- Com os testes, é possível ter uma maior garantia de que o sistema não possui erros críticos, os quais, quando existem, podem causar grandes prejuízos para o cliente;
- Com os testes, novos clientes ficarão interessados em seus sistemas devido às recomendações;
- Quando há uma equipe de teste para avaliar o sistema, todos saem ganhando: o dono da empresa (menos gasto com marketing e mais prestígio para a empresa), o cliente (satisfação e confiança no sistema), os usuários (conseguem usar o sistema de forma útil e correta), os programadores (motivação por criar sistemas com qualidade) e os testadores (mais empregos).

# Objetivos do Teste de Software

- Iniciar um processo para executar um programa com a intenção de descobrir um erro;

- Um bom caso de teste é aquele que apresenta uma elevada probabilidade de revelar um erro ainda não descoberto;

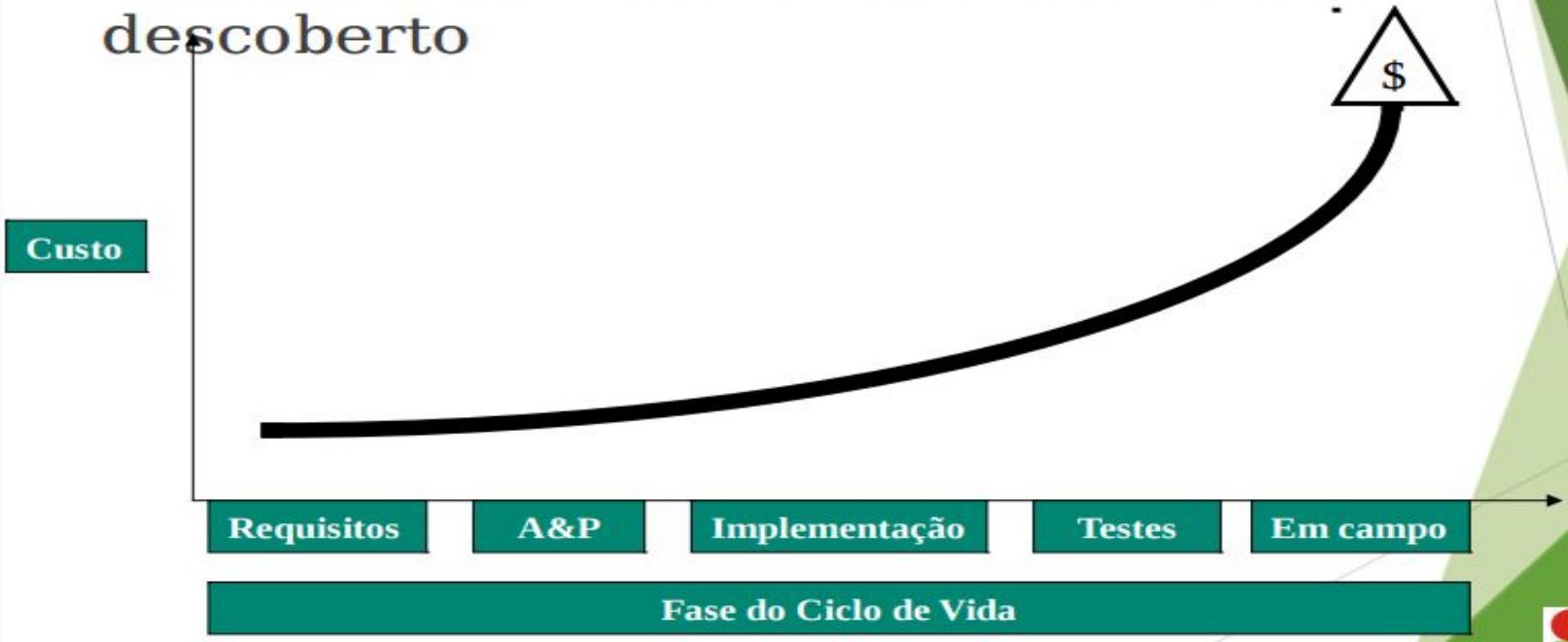
- Um teste bem sucedido é aquele que revela um erro ainda não descoberto.

# Objetivos do Teste de Software

As três regras anteriores expressam o objetivo primordial do teste que é o de encontrar erro, contrariando a falsa ideia de que uma atividade de teste bem-sucedida é aquela em que nenhum erro foi encontrado.

# Custo da Correção de Defeitos de Software

- O custo aumenta exponencialmente, quanto mais tarde no ciclo de vida o defeito for descoberto

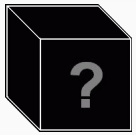



INSTITUTO FEDERAL
PERNAMBUCO

# Técnicas de Testes de Software?

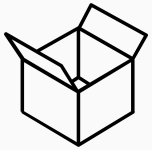
# Técnicas de Testes de Software

**Caixa Preta**




– Conhecendo-se a função específica que um produto projetado deve executar, testes podem ser realizados para demonstrar que cada função é totalmente operacional (teste de caixa preta - “black box”).

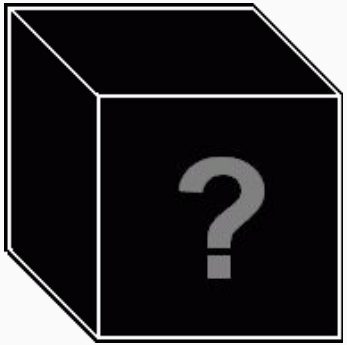
**Caixa Branca**



– Conhecendo-se o funcionamento interno de um produto, testes podem ser realizados para garantir que “todas as engrenagens”, ou seja, que a operação interna de um produto tem um desempenho de acordo com as especificações e que os componentes internos foram adequadamente postos à prova (teste de caixa branca - “white box”).

# Teste Caixa Preta

Teste de caixa preta refere-se aos testes realizados nas interfaces do SW (a entrada é adequadamente aceita e a saída é corretamente produzida com a integridade das informações externas mantida)



# Teste Caixa Branca

Teste de caixa branca baseia-se num minucioso exame dos detalhes procedimentais, através da definição de todos os caminhos lógicos possíveis.

- Infelizmente estes testes apresentam problemas logísticos, uma vez que o número destes possíveis caminhos lógicos pode ser muito grande, o que levaria a um tempo infinito.

- Entretanto este tipo de teste não pode ser desprezado como pouco prático, podendo-se optar por um número limitado de opções

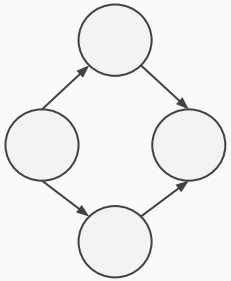
# Tipos de Testes de Software?

# Técnicas Caixa Branca

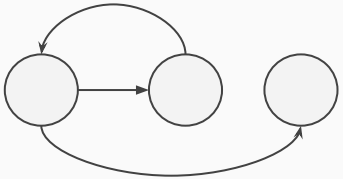
O teste é uma técnica de caixa branca, onde calcula se a complexidade lógica do *software* e utiliza esta medida como base para descobrir os caminhos básicos do *software* e exercendo o teste de modo que todos os caminhos sejam efetuados.

Notação de grafo de fluxo:  notação simples para representação do fluxo de controle, que descreve o fluxo lógico:

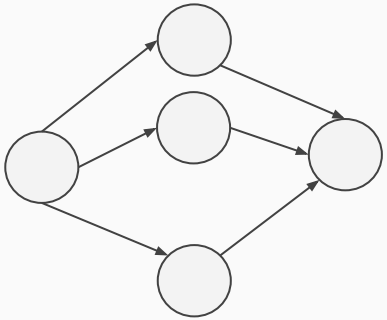
sequência



if



while



case

# Complexidade Ciclomática

É uma métrica de SW que proporciona uma medida quantitativa da complexidade lógica de um programa

- O valor computado da complexidade ciclomática define o número de caminhos independentes do conjunto básico de um programa e oferece-nos um limite máximo para o número de testes que deve ser realizado para garantir que todas as instruções sejam executadas pelo menos uma vez.

# Testes Estáticos

- Os testes estáticos são aqueles realizados sobre o código-fonte do software, utilizando como técnica básica a inspeção visual. Este tipo de teste é de simples implementação.

# Testes Dinâmicos

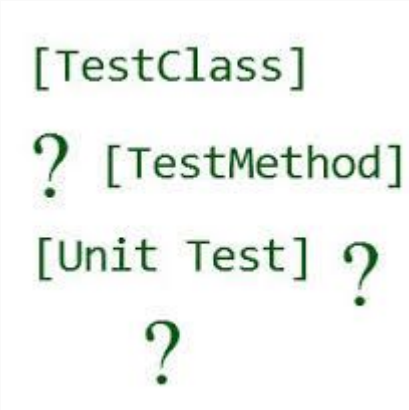
Enquanto a análise estática está focada no código da aplicação, a análise dinâmica se comporta como um hacker tentando encontrar uma vulnerabilidade em tempo de execução da aplicação, tais como:

- SQL Injection
- Campos em formulários ou outro tipo de entrada de dados,
- Manipulação de dados que são passados para a aplicação

Verifica na prática a existência ou não de uma vulnerabilidade.

# Testes de Unidade

Concentra-se no esforço de verificação da menor unidade de projeto de SW - o módulo. Baseia-se quase sempre na técnica de caixa branca (com menor incidência na O.O.) e pode ser realizado em paralelo para múltiplos módulos.



# Teste de Integração

- O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto realizando-se ao mesmo tempo, testes para descobrir erros associados a interfaces (entradas e saídas entre módulos devem se compatibilizar).

# Teste de Validação

São definidas expectativas razoáveis na Especificação de Requisitos de SW, que descreve todos os atributos do SW visíveis ao usuário. A validação é bem-sucedida quando o SW funciona de uma maneira razoavelmente esperada pelo cliente.

# Teste de Regressão

- O teste de regressão é uma técnica do teste de software que consiste na aplicação de versões mais recente do software, para garantir que não surgiram novos defeitos em componentes já analisados.

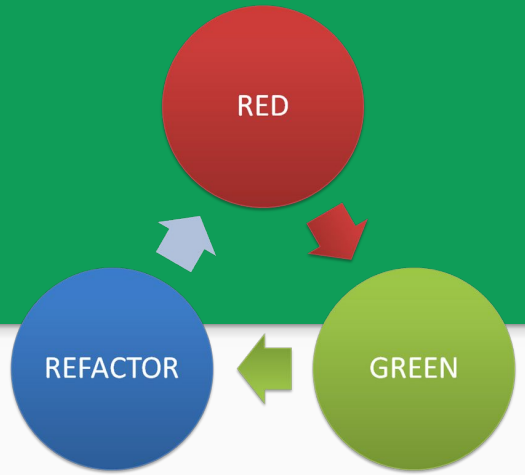
# Teste Alfa


- São os testes de aceitação, feitos pelo usuário, e visam descobrir erros cumulativos que poderiam deteriorar o sistema no decorrer do tempo.
- O teste alfa é executado nas instalações do desenvolvedor, sendo acompanhado pelo desenvolvedor, que registra os problemas encontrados no uso.

# Teste Beta


- O Teste Beta é realizado em uma ou mais instalações do cliente pelo usuário final do software. Geralmente o desenvolvedor não está presente.
- É uma aplicação real do software, sem que haja controle por parte do desenvolvedor. Os problemas são registrados pelo usuário e repassados regularmente ao desenvolvedor.

# TDD




- Desenvolvimento guiado pelos testes
  - Só escreva código novo se um teste falhar
  - Refatore até que o teste funcione
  - Alternância: "red/green/refactor" -

nunca passe mais de 10 minutos sem que a barra do JUnit fique verde.

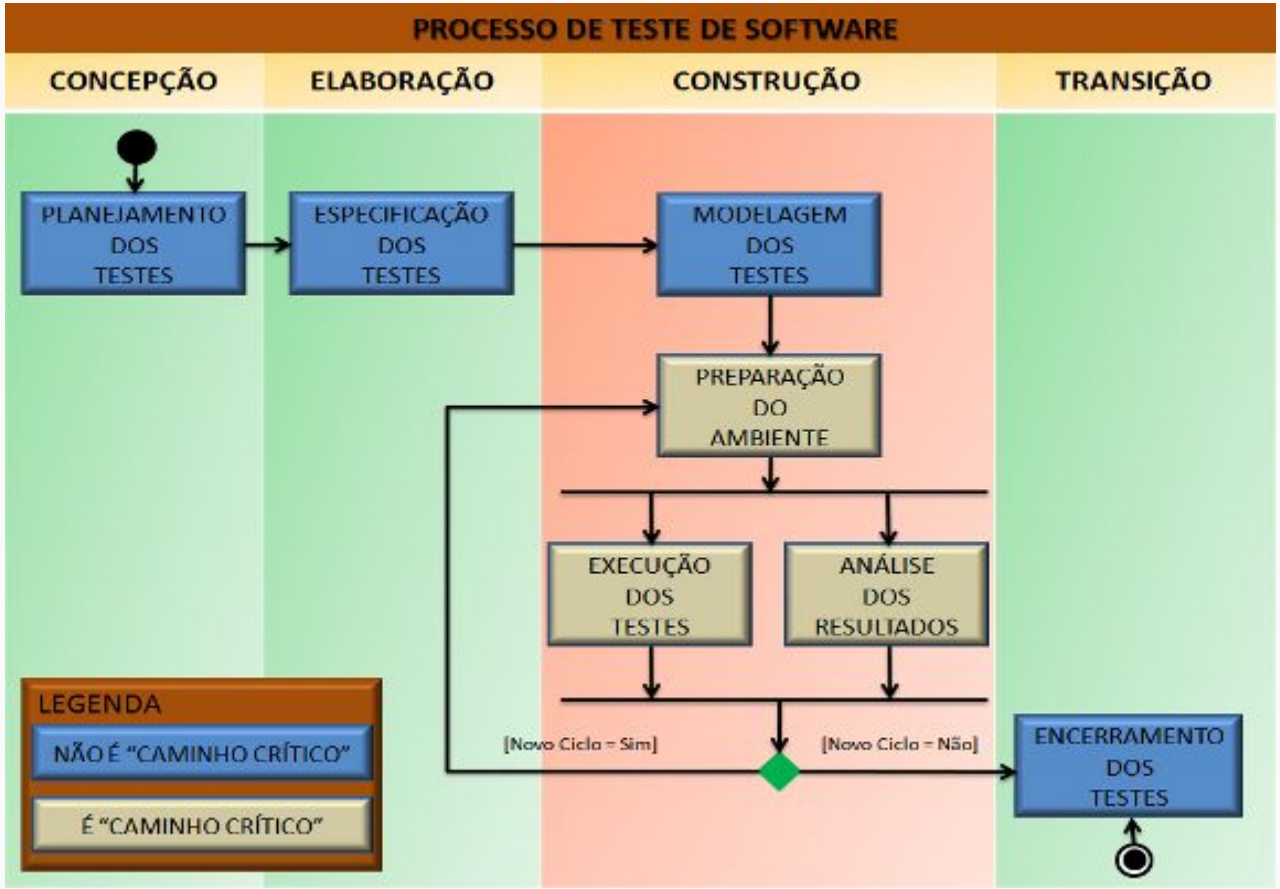
# Teste de Sistemas

- É uma série de diferentes testes, cujo propósito primordial é pôr completamente à prova o sistema baseado em computador.

- **Teste de recuperação**: é um teste de sistema que força o SW a falhar de diversas maneiras e verifica se a recuperação é adequadamente executada.
- **Teste de segurança**: tenta verificar se todos os mecanismos de proteção embutidos em um sistema o protegerão, de fato, de acessos indevidos.
- **Teste de estresse**: executa o sistema de uma forma que exige recursos em quantidade. Essencialmente o analista tenta destruir o programa.
- **Teste de desempenho**: é idealizado para testar o desempenho de “runtime” do SW dentro do contexto de um sistema integrado.

# Processo de Testes de Software

# Processo de Teste de Software (Visão do RUP)



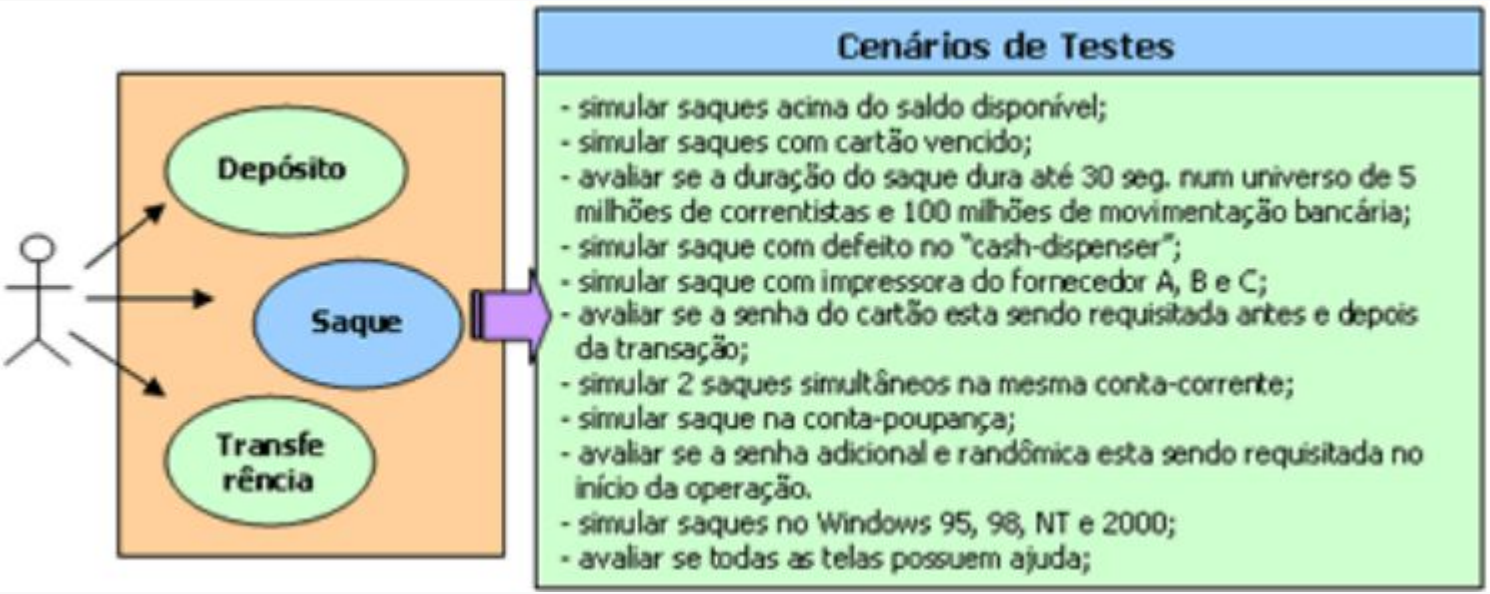
# Planejamento de Teste

- Definição de uma proposta de testes baseada nas expectativas do Cliente em relação à :
  - prazos,
  - custos
  - qualidade esperada

- Possibilidade de dimensionar a equipe e estabelecer um esforço de acordo com as necessidades apontadas pelo Cliente.

# Especificação dos Testes

- Identificação dos casos de testes que deverão ser construídos e/ou modificados em função das mudanças solicitadas pelo Cliente.



# Especificação dos Testes (Categorias)

| Funcional | Segurança | Usabilidade | Performance |
|---|---|---|---|
| - simular saques acima do saldo disponível;<br>- simular saque na conta-poupança;<br>**- simular saque acima do valor do limite da conta;**<br>**- simular saque com valores não múltiplos das notas;**<br>**- simular saque com valores não múltiplos das notas;** | - simular saques com cartão vencido;<br>- avaliar se a senha do cartão esta sendo requisitada antes e depois da transação;<br>- avaliar se a senha adicional e randômica esta sendo requisitada no início da operação;<br>**- simular saque noturno acima do valor permitido;** | - avaliar se todas as telas possuem ajuda;<br>**- avaliar se mensagens são claras e objetivas;**<br>**- avaliar se o padrão visual é mantido em todos os momentos;**<br>**- avaliar se todas as operações possuem caminhos de fuga;** | - avaliar se a duração do saque dura até 30 seg. num universo de 5 milhões de correntistas e 100 milhões de movimentação bancária;<br>**- garantir que manipulação com dispositivos físicos no saque não ultrapassem 10 seg. da operação;** |
| **Carga e Concorrência** | **Configuração** | **Recuperação** | **Contingência** |
| - simular 2 saques simultâneos na mesma conta-corrente;<br>**- simular 10.000 saques simultâneos;** | - simular saque com impressora do fornecedor A, B e C;<br>- simular saques no Windows 95, 98, NT e 2000;<br>**- simular saque com impressora do fornecedor X, Y e Z;** | - simular saque com defeito no "cash-dispenser;<br>**- simular saque com defeito na impressora;**<br>**- simular saque com falha de conexão com a central;**<br>**- simular saque com queda de energia;** | **- disparar processo de instalação emergencial;** |

# Especificação de Teste (Caso de Teste)

- Em engenharia de software, caso de teste é um conjunto de condições usadas para teste de software. Ele pode ser elaborado para identificar defeitos na estrutura interna do software.
- Podemos utilizar a ferramenta de casos de uso para criar e rastrear um caso de teste, facilitando assim identificação de possíveis falhas.

# Exemplo de Caso de Teste

| | |
|---|---|
| ID | CT-001 |
| Caso de Teste: | Efetuar login no sistema. |
| Funcionalidade: | Login |
| Pré - Condição: | 1. Possuir usuário válido para efetuar o login. |
| Procedimento: | 1. Acessar o sistema;<br>2. Inserir usuário e senha;<br>3. Clicar em "Login". |
| Resultado Esperado: | 1. Login do usuário efetuado com sucesso. |

| | |
|---|---|
| ID | CT-002 |
| Caso de Teste: | Cadastrar cliente no Sistema |
| Funcionalidade: | Cadastro de cliente |
| Pré - Condição: | Estar logado no sistema |
| Procedimento: | 1. Entrar na tela "Cadastro" -> "Cliente"<br>2. Inserir dados do cliente.<br>3. Clicar em "Salvar". |
| Resultado Esperado: | 1. Cliente é cadastrado com sucesso.. |

# Exemplo de Caso de Teste

| Test Case Tradicional | |
|---|---|
| **ID** | TCS-123 |
| **Nome** | Consultar um Triângulo Escaleno |
| **Ambiente** | RVM-1.8.7 |
| **Ator** | Aluno de Matemática |
| **Pré-Condições** | Não existir nenhum triângulo |
| **Procedimentos (Entradas e Saídas)** | 1 – Abra a aplicação<br>2 – Solicite consultar o tipo de um triângulo<br>3 – O sistema solicita os tamanhos dos lados do triângulo<br>4 – Informe o lado A como “3“<br>5 – Informe o lado B como “4“<br>6 – Informe o lado C como “5“<br>7 – O sistema informa que o triângulo é “Escaleno” |
| **Pós-Condições** | Existe um triângulo do tipo escaleno |

# Modelagem dos Testes

Identificação de todos os elementos necessários para a implementação de cada caso de teste especificado:

– modelagem das massas de testes

– definição dos critérios de tratamento de arquivos (descaracterização e comparação de resultados).

# Preparação do Ambiente

- Conjunto de atividades que visa a disponibilização física de um ambiente de testes para sofrer a bateria de testes planejadas nas etapas anteriores de forma contínua e automatizada (sem intervenção humana).

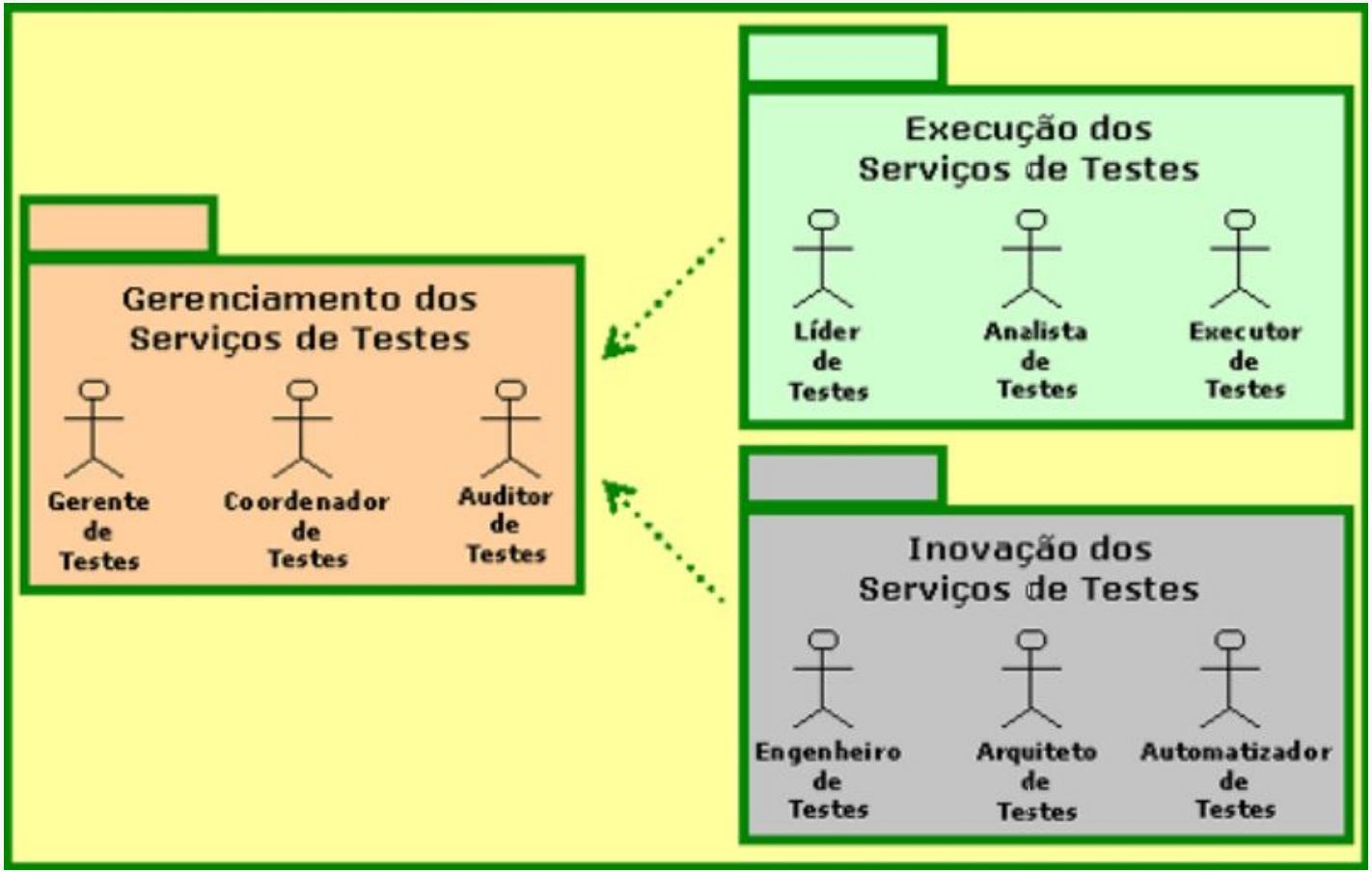
# Execução dos Testes

- Execução e conferência dos testes planejados, de forma a garantir que o comportamento do aplicativo permanece em "conformidade" com os requisitos contratados pelo Cliente.

# Análise dos Resultados

- Análise e confirmação dos resultados relatados durante a fase de execução dos testes.
- Os resultados em "não-conformidade" deverão ser "confirmados" e "detalhados" para que a Fábrica de Software realize as correções necessárias.
- Já os em "conformidade" deverão ter seu resultado "POSITIVO" reconfirmado.

# Equipes de Testes




Dúvidas ?