

# *Testes de Software*



Prof<sup>a</sup> Flavia Garcia  
Senac Rio

# Competências

## **Competência:**

- Planejar e implementar testes de software.

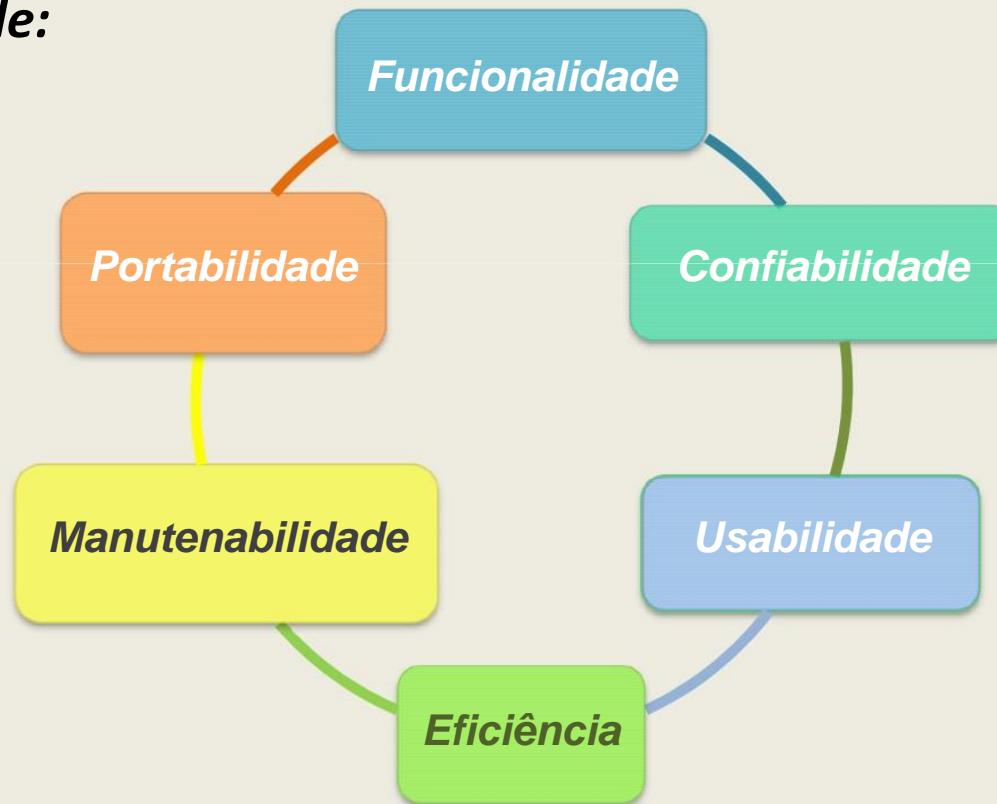
## **Bases tecnológicas (conteúdos):**

- Fundamentos de Teste de Software
- Planejamento de Testes de Software
- Processos de Teste de Software
- Técnicas de Validação de Requisitos para Teste de Software
- Técnicas de Testes de Software Estruturais e Funcionais
- Cenários de Teste
- Introdução a Automação de Testes de Software
- Controle de Falhas
- Relatórios de Teste de Software
- Principais Indicadores de Testes de Software

# *Como adquirir qualidade em um software?*



**A Norma ISO 9126 define as seguintes características para qualidade:**



# Introdução

## *O que é teste de software?*

Os testes são realizados com a intenção de descobrir erros e defeitos em um sistema. [Myres, 2004]

Os testes de software podem ser usados para mostrar a presença de defeitos, mas nunca para mostrar a ausência deles. [Dijkstra, 1972]

Os testes de software servem para medir a confiabilidade de um sistema: à medida que poucos defeitos são encontrados em um determinado tempo, o software é considerado mais confiável.

# ***Por que testar é necessário?***

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.

# **Conceitos Básicos de Teste**

## **Artefatos de Teste**

- *todo o conjunto de documentação gerado pelo processo de teste de software.*

## **Caso de Teste**

- *é composto por um conjunto de entradas, por passos de execução e um resultado esperado.*

## **Roteiro de Teste**

- *É composto por um conjunto de casos de teste definidos para uma determinada especificação.*

***Requisitos***

- *regras de negócio do sistema.*

***Testar***

- *descobrir falhas através da execução do sistema.*

***Bug***

- *é um defeito encontrado no sistema em execução.*

## *Confiabilidade do Software*

*Confiabilidade do Software é a probabilidade que o software não causará uma falha no sistema por um tempo especificado, sob condições determinadas.*

## *O custo de um defeito*

*O custo da correção de um defeito tende a ser cada vez maior quanto mais tarde ele for descoberto. [Myres, 2004]*

# Fundamentos



Você acha  
importante realizar  
testes em um sistema  
que está sendo  
desenvolvido??

Não acredito que seja  
necessário

Nem considero entregar um  
software sem testar

# Fundamentos

Se não testarmos o sistema, como iremos garantir que cada item do projeto seja atendido conforme especificado?



Exemplo de itens que devem ser verificados no projeto

Requisitos Funcionais

Requisitos Não Funcionais

Regras de Negócio

Casos de uso

# Por que realizar testes no software?

A cartoon illustration featuring a boy with glasses and a purple cap, and a man sitting cross-legged on a blue and green globe while working on a laptop. The boy is speaking into a speech bubble, and three numbered callouts provide reasons for testing.

Por que testar ?!?  
veja alguns  
motivos:

- 1 **Economia:**  
Reduz o tempo  
gasto com  
retrabalho  
relacionado às  
manutenções  
corretivas.
- 2 **Qualidade:**  
Os testes ajudam a  
garantir que o  
produto atendeu  
todas as  
especificações.
- 3 **Negócio:**  
Falhas percebidas  
pelo cliente são um  
risco ao negócio.  
Testes podem  
identificá-las a  
tempo.

Sem dúvida o impacto do  
desenvolvimento de  
software sem a correta  
aplicação de testes pode  
deixar o cliente bem  
insatisfeito...

# Conceitos incorretos

É importante definir corretamente Teste de Software!

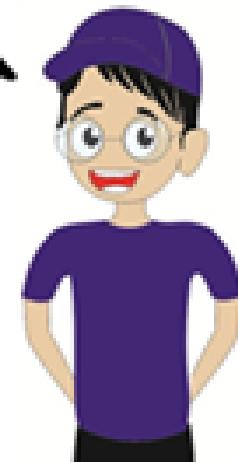
Uma visão distorcida deste conceito pode levar a um processo de testes ineficiente.

Veja:



O teste de software é o processo de demonstrar que os defeitos não estão presentes.

Errado! O que queremos com o teste é justamente encontrar erros.





Se o testador tiver por objetivo provar que o programa funciona, corre o risco de ser tendencioso na escolha dos casos de teste, ignorando situações que poderiam apontar defeitos.

Teste de software é a atividade de executar um programa ou sistema com a intenção de encontrar defeitos

Myers, 1979

Veja alguns conceitos mais adequados:

Conferir se o software está fazendo o que deveria fazer, de acordo com os seus requisitos, e não está fazendo o que não deveria fazer



Fonte:

NETO, Michele. Vídeo Aula1 – Conceitos iniciais em  
teste de Software. Disponível em:  
<http://www.youtube.com/watch?v=7kec8BT8gU4>.

# Terminologia



# *Erro, Defeito e Falha*

**Erro:** é uma ação humana que produz um resultado incorreto.

**Defeito:** A manifestação de um erro no software.

Também conhecido como Bug

**Falha:** quando o sistema se comporta de forma inesperada devido ao defeito.

# *Erro, Defeito e Falha*



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

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

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



# Terminologia – *Bug (Curiosidade)*



O uso da palavra “Bug” na informática surgiu por causa de um problema em um supercomputador da Universidade de Harvard (EUA), nos anos de 1960. A máquina parou de funcionar e os técnicos não conseguiram descobrir a causa. Quando desmontaram o computador, encontraram um inseto (bug, em inglês) morto no meio dos circuitos, interrompendo a corrente elétrica.

## Curiosidade BUG...



### ENIAC

O ENIAC (Electronic Numerical Integrator and Computer), primeiro computador digital completamente eletrônico, contribuiu para o uso da palavra "BUG"

3

# Terminologia



Mas afinal, qual é a diferença entre engano, defeito, erro e falha?

O foguete Ariane V explodiu em 1996. O software de controle do voo indicou uma direção errada ao foguete (**esta foi a falha**) devido a uma conversão incorreta de uma variável tipo real de 64 bits em um inteiro de 16 bits (**este foi o defeito**). Esta conversão produziu um erro de overflow, isto é, um valor errôneo de uma variável (**este foi o erro**), que ocasionou a resposta incorreta do software. O **engano** foi, provavelmente, cometido por um programador, que inseriu no software um comando de atribuição indevido.



## Engano



Ação humana que produz um resultado incorreto, como uma ação incorreta tomada pelo programador.

## Defeito

Passo, processo ou definição de dados incorretos, como por exemplo, uma instrução ou comando incorreto.



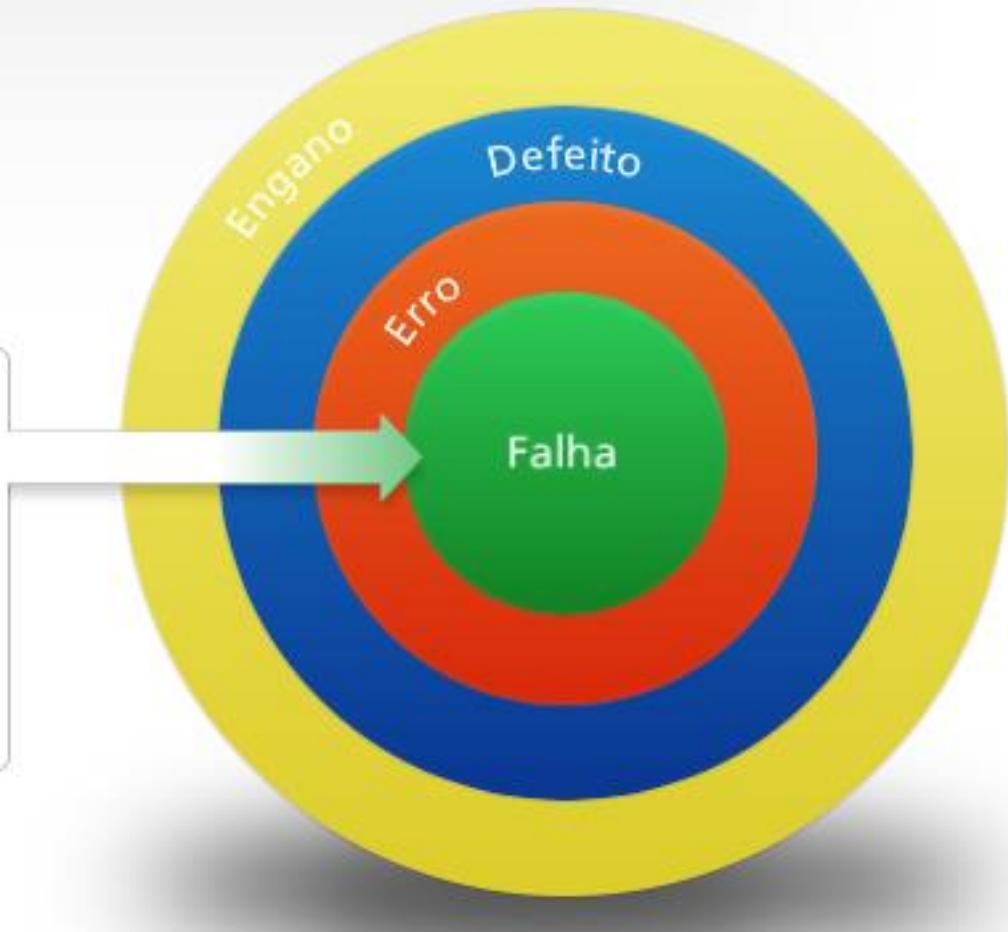
## Erro

Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro.



## Falha

Produção de uma saída incorreta com relação à especificação. A falha é a percepção do erro.



# Resumindo ...

Teste de software é uma das atividades do processo de desenvolvimento de sistema de software que visa executar um programa de modo sistemático com o objetivo de encontrar falhas. Perceba que isto requer verificação e validação de software. Nesse sentido, definir quando as atividades de verificação e validação iniciam e terminam, como os atributos de qualidade serão avaliados e como os *releases* do software serão controlados, são questões que devem ser acompanhadas ao longo do processo de software.

Vale ressaltar que teste não deve ser a última atividade do processo de desenvolvimento de software. Ela ocorre durante todo o processo, como exemplificado na visão geral do processo RUP (*Rational Unified Process*).

# Resumindo ...

## **Engano:**

Ação humana que produz um resultado incorreto, como uma ação incorreta tomada pelo programador.

## **Defeito:**

Passo, processo ou definição de dados incorretos, como por exemplo, uma instrução ou comando incorreto.

## **Erro:**

Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro.

## **Falha:**

Produção de uma saída incorreta com relação à especificação.

A falha é a percepção do erro.

**Algumas terminologias  
que não podemos  
esquecer...**



O estudo de testes de software geralmente é organizado em três categorias:

Técnicas de Teste

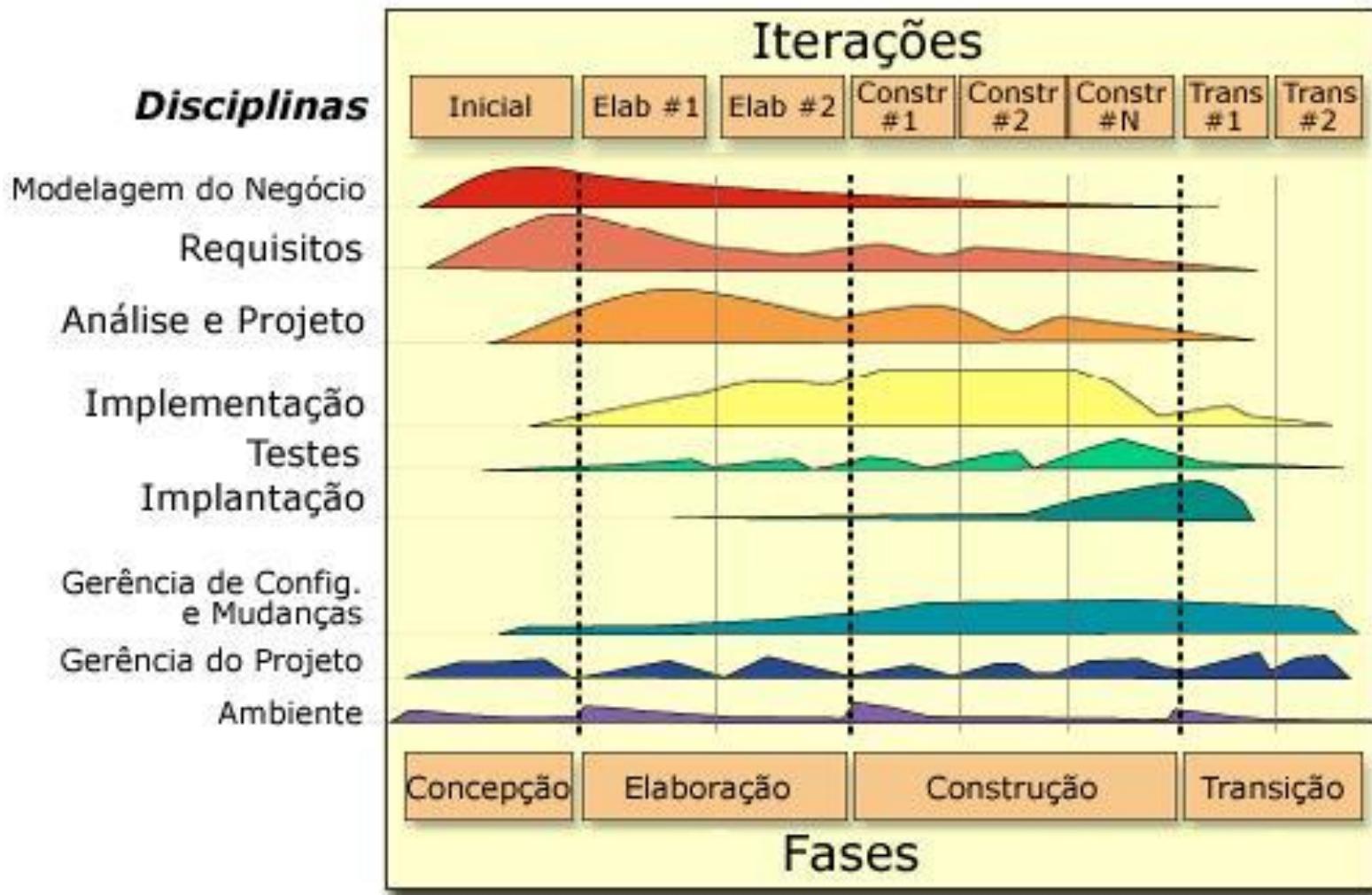


Níveis de Teste



Tipos de Teste





# Planejamento de Testes

O plano de teste é um dos documentos produzidos na condução de um projeto. Ele funciona como:

- § Um ‘integrador’ entre diversas atividades de testes no projeto;
- § Mecanismo de comunicação para os stakeholders (i.e. a equipe de testes e outros interessados);
- § Guia para execução e controle das atividades de testes.

O plano de teste, que pode ser elaborado pelo gerente de projeto ou gerente de testes, visa planejar as atividades a serem realizadas, definir os métodos a serem empregados, planejar a capacidade necessária, estabelecer métricas e formas de acompanhamento do processo. Nesse sentido, deve conter:

- § Introdução com identificação do projeto (definições, abreviações, referências), definição de escopo e objetivos;
- § Conjunto de requisitos a serem testados;
- § Tipos de testes a serem realizados e ferramentas utilizadas;
- § Recursos utilizados nos testes;
- § Cronograma de atividades (e definição de marcos de projeto).

# Planejamento de Testes

- O Plano de Testes é o documento responsável por apresentar o planejamento para execução dos testes do software em desenvolvimento, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste.
- Tendo como referência a norma IEEE 829, padrão internacional que trata da documentação de teste de software, verifica-se a necessidade das seguintes informações em um plano de testes:

# Planejamento de Testes

**Em outras palavras, um plano de teste deve definir:**

1. Os itens a serem testados: o escopo e objetivos do plano devem ser estabelecidos no início do projeto.
2. Atividades e recursos a serem empregados: as estratégias de testes e recursos utilizados devem ser definidos, bem como toda e qualquer restrição imposta sobre as atividades e/ou recursos.
3. Os tipos de testes a serem realizados e ferramentas empregadas: os tipos de testes e a ordem cronológica de sua ocorrência são estabelecidos no plano.
4. Critérios para avaliar os resultados obtidos: métricas devem ser definidas para acompanhar os resultados alcançados.

Perceba que o planejamento é necessário a fim de antecipar o que pode ocorrer e, portanto, provisionar os recursos necessários nos momentos adequados. Isto significa coordenar o processo de teste de modo a perseguir a meta de qualidade do produto (sistema de software).

# Planejamento de Testes

## Exemplificando o Plano de Teste

O plano de teste contém um conjunto de informações que permite ao gerente de projeto não apenas coordenar as atividades de testes de um projeto, mas também monitorar seu progresso e verificar se o executado está em conformidade com o planejado.

A **Tabela** apresenta uma relação dos itens consideradas imprescindíveis em um plano de teste. A relação de itens não pressupõe a intenção de ser completo, mas de apontar os itens considerados como obrigatórios num plano de teste de uma instituição.

# Planejamento de Testes

| <b><i>Itens de um Plano de Teste</i></b>     | <b><i>Conteúdo</i></b>                                                                                                                                                                                                                                                              |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>1. Introdução</i></b>                  | Contém uma identificação do projeto, descrição dos objetivos do documento, o público ao qual ele se destina e escopo do projeto a ser desenvolvido. Pode adicionalmente conter termos e abreviações usadas, além de informar como o plano deve evoluir.                             |
| <b><i>2. Requisitos a serem testados</i></b> | Esta seção descreve em linhas gerais o conjunto de requisitos a serem testados no projeto a ser desenvolvido, comunicando o que deve ser verificado. Exemplos de requisitos a serem testados são: desempenho, segurança, interface de usuário, controle de acesso, funcionalidades. |

# Planejamento de Testes

|                                                     |                                                                                                                                                                                                                                                                                                         |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>3. Estratégias e ferramentas de teste</b></p> | <p>Apresenta um conjunto de tipos de testes a serem realizados, respectivas técnicas empregadas e critério de finalização de teste. Além disso, é listado o conjunto de ferramentas utilizadas.</p>                                                                                                     |
| <p><b>4. Equipe e infra-estrutura</b></p>           | <p>Contém descrição da equipe e da infra-estrutura utilizada para o desenvolvimento das atividades de testes, incluindo: pessoal, equipamentos, software de apoio, materiais, dentre outros. Isto visa garantir uma estrutura adequada para a execução das atividades de testes previstas no plano.</p> |

# Planejamento de Testes

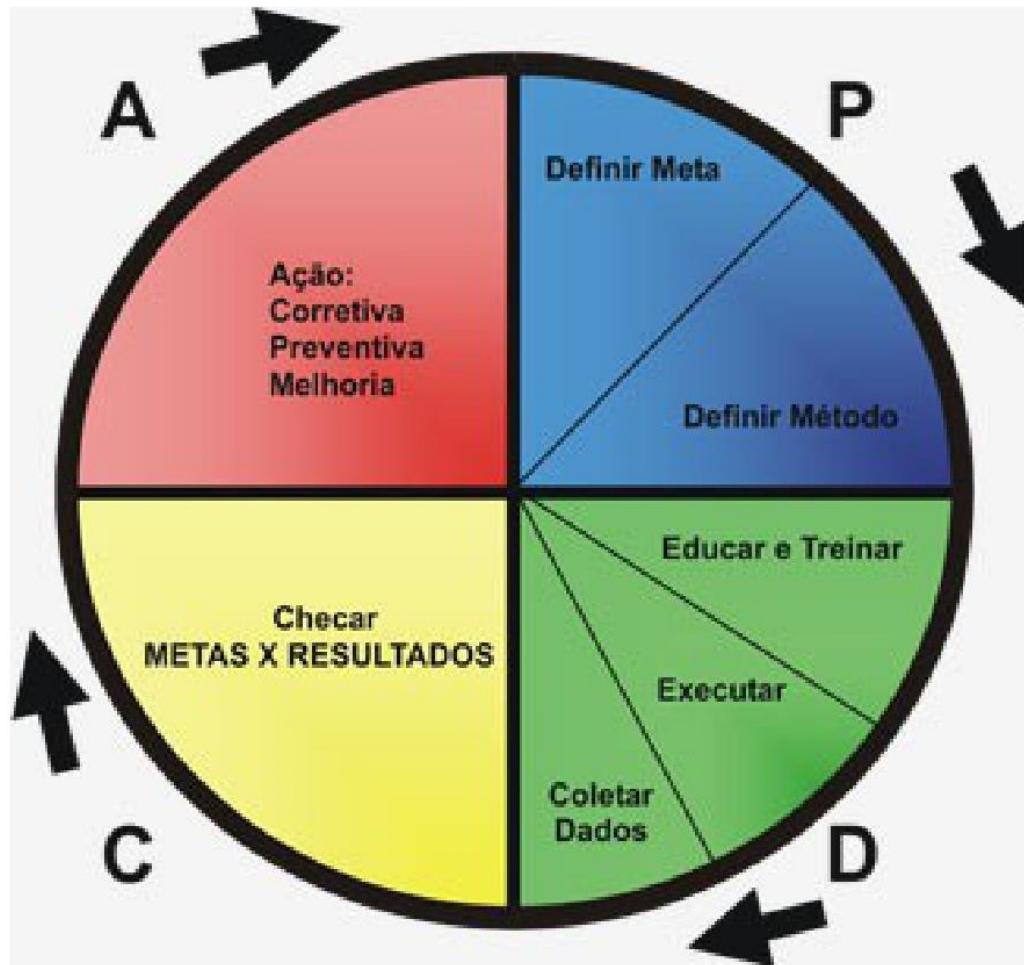
## *5. Cronograma de atividades*

Contém uma descrição de marcos importantes (*milestones*) das atividades (incluindo as datas de início e fim da atividade). Apenas marcos relevantes devem ser listados, ou seja, aqueles que contribuirão nas atividades de testes. Por exemplo: projeto de testes, execução de testes ou avaliação de testes.

## *6. Documentação complementar*

Apresenta-se uma relação dos documentos pertinentes ao projeto.

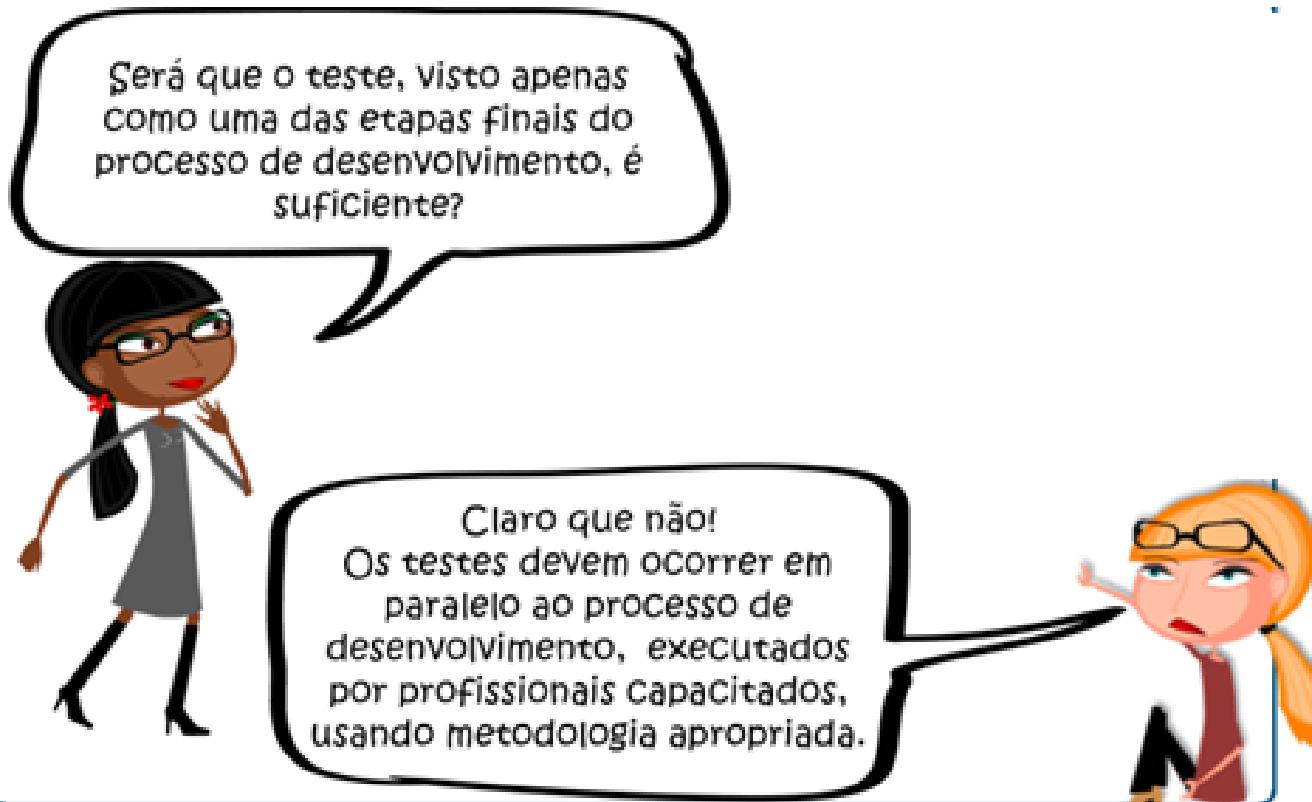
# Processo de Teste - PDCA



# *Entendendo o Processo de Teste*

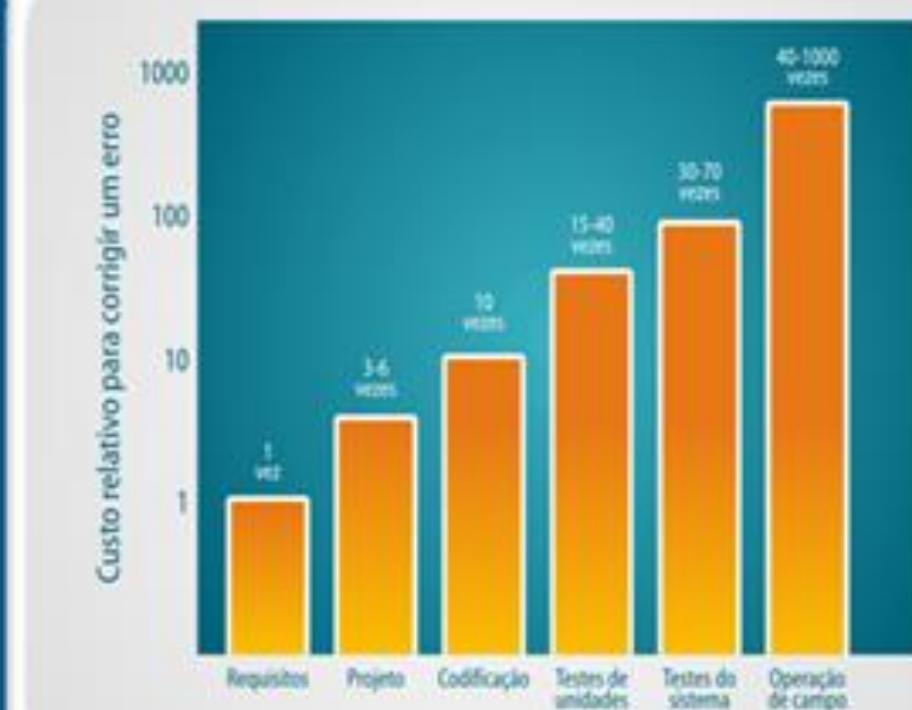


# *Entendendo o Processo de Teste*



# *Entendendo o Processo de Teste*

O projeto de teste de software deve começar paralelamente ao projeto de desenvolvimento, pois quanto antes o defeito for encontrado, menor será o custo de sua correção. Essa realidade é comprovada pela Regra 10 de Myers, que estabelece que o custo de correção de defeitos tende a aumentar quanto mais tarde o defeito é detectado.



# Processo de Teste de Software

*Um processo de teste básico deve contemplar:*

- Planejamento e controle;

- Análise **e** modelagem;

- Implementação e execução;

- Avaliação de critérios de saída e relatórios;

- Atividade de encerramento dos testes.

- Melhoria/Corretiva/Preventiva.

# Processo de Teste de Software (IEEE 12207)

- Prover uma metodologia de trabalho, ferramentas, e profissionais de forma a colaborar com o processo de desenvolvimento de software da organização, onde:

# Principais Funções :

## Quem participa do processo de teste?

### Introdução



Durante o desenvolvimento de um software várias pessoas trabalham para garantir que o projeto seja concluído com êxito, a essas pessoas são atribuídos papéis que definem o que cada uma deverá fazer dentro do processo de desenvolvimento.

## Quem participa do processo de teste?

### Gestor de qualidade

Durante o ciclo de vida do desenvolvimento do software, eu respondo pelas ações de controle de qualidade dos produtos, ou seja, se estes foram desenvolvidos de acordo com as especificações acordadas e se atendem às expectativas dos usuários (ou clientes).



## Quem participa do processo de teste?



### Analista de sistemas

Como Analista de Sistemas acumulo maior número de responsabilidades no ciclo de desenvolvimento de software.

Minhas atividades são:

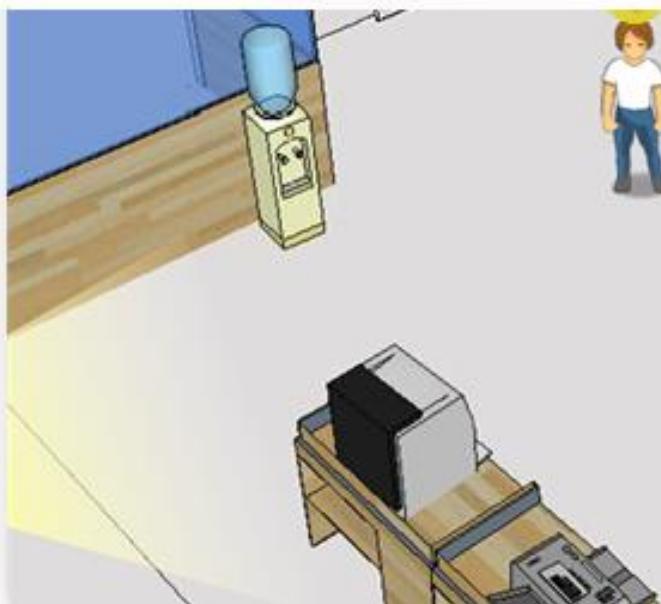
- Levantar e documentar os requisitos;
- Apoiar atividades de planejamento do projeto;
- Modelagem do software;
- Preparar e manter a documentação do projeto;
- Acompanhar as atividades de construção dos testes unitários;
- Planejar e executar os teste de integração.



Qu

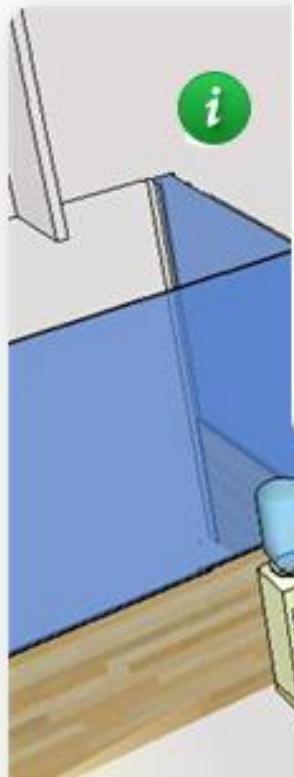
## Programador

Como programador sou responsável pela codificação dos programas conforme a modelagem definida pelo analista. Porém, a construção dos testes de unidade também são minha responsabilidade, assim como a realização dos testes de integração.



## Quem participa do processo de teste?

### Líder do projeto de teste



**i**

Sou o técnico responsável pela liderança de um projeto de teste específico, normalmente relacionado a um sistema.



**4**

**5**

**6**

**7**

## Quem participa do processo de teste?

### Arquiteto de teste

Meu trabalho como arquiteto diz respeito à organização da infra-estrutura de teste:

- Ambiente de teste;
- Ferramentas e Capacitação da equipe.



## Quem participa do processo de teste?



## Quem participa do processo de teste?



## Quem participa do processo de teste?



### Gerente de Projeto

Meu trabalho é conduzir o projeto tomando providências necessárias para que o software seja construído dentro do prazo e custo determinado, também é minha tarefa assegurar que toda documentação do sistema seja preparada para permitir manutenções futuras.

Como gerente de projeto devo seguir e fazer cumprir a Metodologia de Desenvolvimento de Sistemas(MDS) adotada pela organização.



Vamos exercitar?



Qual definição para teste de software está correta conforme o conceito definido por Myers?

- O objetivo do teste é mostrar que um programa executa corretamente as funções para as quais foi destinado.
- Testar é o processo de criar a confiança de que um programa faz aquilo que se supõe que ele faça.
- Testar é o processo de execução de um programa com a intenção de encontrar erros.
- O teste é o processo de demonstrar que os erros não estão presentes

Quais afirmações abaixo não são definidas pela regra 10 de Myers:

- O custo para correção de um defeito aumenta de forma diretamente proporcional a complexidade do software.
- O custo para correção de um defeito aumenta de forma diretamente proporcional ao tempo de demora em sua localização.
- O custo para correção de um defeito aumenta de forma diretamente proporcional ao tamanho do software.
- O custo para correção de um defeito aumenta de forma inversamente proporcional ao tamanho da equipe.

No processo de teste os casos de teste são:

- Elaborados pelo líder do projeto de teste e executados pelo testador
- Elaborados e executados pelo testador
- Elaborados pelo arquiteto de testes e executados pelo testador
- Elaborados pelo analista de testes e executados pelo testador

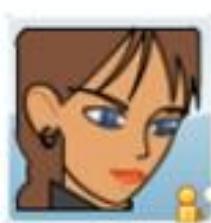
Entre os papéis relacionados no processo de desenvolvimento, clique sob quem se preocupa em assegurar que custos e prazos estipulados serão cumpridos.



**Analista de  
Sistemas**



**Gestor de  
Qualidade**



**Gerente de  
Projetos**



**Arquiteto  
de Testes**

O processo de desenvolvimento e de testes devem coexistir. Fato pelo qual, alguns dos profissionais participam das duas atividades, como por exemplo: "Além de elaborar requisitos e modelar o sistema, também participa da construção de testes unitários e execução de testes de integração". De quem estamos falando?

- Analista de Sistemas
- Analista de Testes
- Gerente de Projetos
- Programador

Relacione as colunas segundo o conceito de Engano, Defeito, Erro e Falha.

Engano

Um comportamento inesperado do programa percebido pelo usuário

Defeito

Algo que pode acarretar um comportamento inesperado.

Erro

A diferença entre o resultado que se esperava de um processo e o resultado que ele apresenta

Falha

Um equívoco ou um descuido no momento da codificação do software.

Relacione as colunas segundo o conceito de Engano, Defeito, Erro e Falha.

1

Engano

2

Defeito

3

Erro

4

Falha

4

Um comportamento inesperado do programa percebido pelo usuário

2

Algo que pode acarretar um comportamento inesperado.

3

A diferença entre o resultado que se esperava de um processo e o resultado que ele apresenta

1

Um equívoco ou um descuido no momento da codificação do software.

Segundo a terminologia apresentada pela IEEE 610.12-1990, podemos concluir que:

- Um engano é ocasionado por uma ou mais falhas
- Um defeito, nesse contexto, é sinônimo de falha.
- Uma falha é ocasionada por um ou mais defeitos.
- Uma falha, nesse contexto, é sinônimo de defeito.

Uma vez que defeitos podem ser inseridos no software durante todo seu processo de desenvolvimento, é importante que as atividades de teste ocorram desde o início do projeto. Mas, é possível fazer algum tipo de teste antes de termos, ao menos, uma versão estável para executar?

- Não, os testes dependem de uma versão estável do programa, ou ao menos parte dele, para execução das atividades de teste.
- Sim, durante a definição de requisitos e modelagem do futuro sistema, já é possível realizar atividades inspeção e análise da qualidade dos artefatos. Esta atividade recebe o nome de "Teste Estático"
- Os testes de unidade, são testes estáticos, que não precisam uma versão estável do sistema em execução.
- Os testes servem para conferir se o sistema atende às necessidades do cliente. Portanto, não existem testes sem o programa implementado para testar.

Sobre as atividades de Verificação e Validação (V&V), podemos afirmar que:

- Verificação constitui um teste estático (exige a execução do produto em teste).
- Verificação constitui um teste dinâmico (exige a execução do produto em teste)
- Validação constitui um teste dinâmico (exige a execução do produto em teste).
- Validação constitui um teste estático (não exige a execução do produto em teste).

# Praticando

Em grupo de 4 alunos (2 desenvolvedores e 2 usuários), simular uma reunião para especificação de requisitos de um Sistema de Controle de Biblioteca

Ao término da “reunião”, crie um documento com:

- Nome do sistema
- Áreas envolvidas
- Objetivos do sistema
- Restrições
- Descrição funcional

# Praticando

Em grupo de 4 alunos (2 desenvolvedores e 2 usuários), simular uma reunião para especificação de requisitos de um Sistema de Controle de Biblioteca

Ao término da “reunião”, crie um plano de testes com:

- Nome do sistema
- Áreas envolvidas
- Objetivos do sistema
- Restrições
- Descrição funcional

Entre outros...

# Praticando

Em grupo de 4 alunos (2 desenvolvedores e 2 usuários), desenvolver um plano de testes completo para :

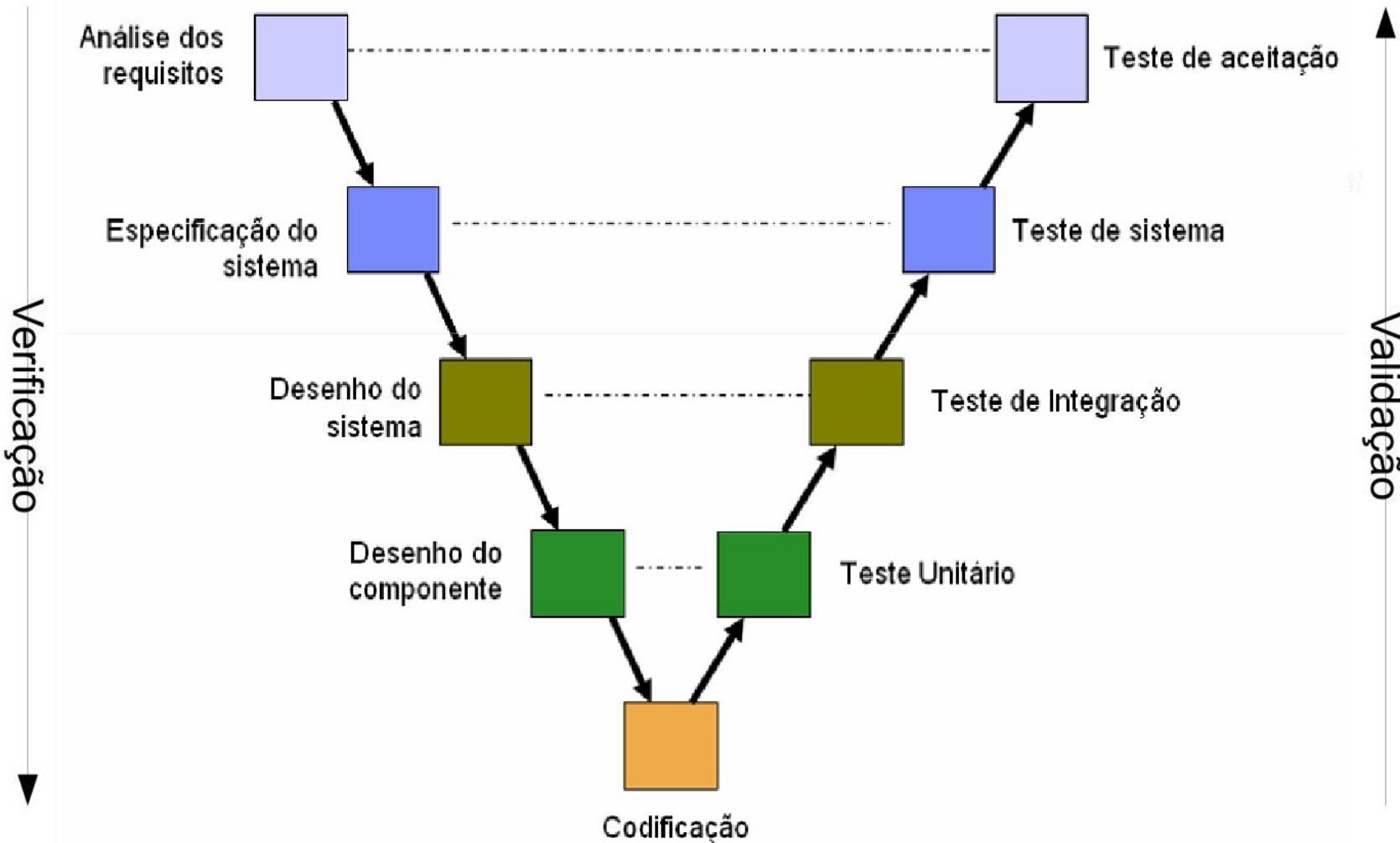
A empresa JAVAS MODAS vende um software de gerenciamento de E-commerce Varejistas.

O cliente necessita de um sistema web/mobile que proporcione maior lucratividade e gestão em suas vendas de forma online, onde seja possível efetuar compras, gestão comercial e estoque.

O Sistema permitirá os gerenciamentos dos Usuário e Produtos, como cadastrar, alterar, pesquisar e excluir. Possibilitará uma listagem dos produtos disponíveis no estoque, realizar vendas e gerar seus relatórios.

# *Verificação e Validação*

## *Processo em “V”*

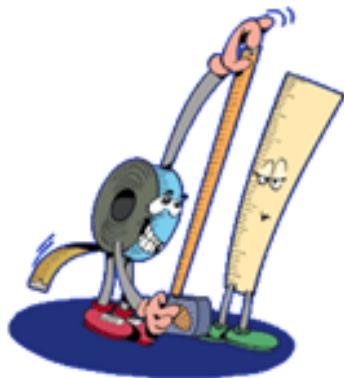


# *Níveis de Teste de Software*

| Atributos        | Nível dos Testes       |                                               |                                         |                                                  |
|------------------|------------------------|-----------------------------------------------|-----------------------------------------|--------------------------------------------------|
|                  | Testes Unitários       | Testes de Integração                          | Testes de Sistema                       | Testes de Aceitação                              |
| Escopo           | <i>Unidades</i>        | <i>Conjunto de unidades agrupadas</i>         | <i>Sistema todo</i>                     | <i>Sistema todo</i>                              |
| Equipe           | <i>Desenvolvedores</i> | <i>Desenvolvedores e Analistas de Sistema</i> | <i>Analista de Testes e Testadores</i>  | <i>Analista de Testes, Testadores e Usuários</i> |
| Origem dos dados | <i>Criação manual</i>  | <i>Criação manual</i>                         | <i>Criação automática / dados reais</i> | <i>Dados reais</i>                               |
| Volume dos dados | <i>Pequeno</i>         | <i>Pequeno</i>                                | <i>Grande</i>                           | <i>Grande</i>                                    |
| Interfaces       | <i>Não existem</i>     | <i>Não existem</i>                            | <i>Simuladas / Reais</i>                | <i>Reais</i>                                     |
| Ambientes        | <i>Desenvolvimento</i> | <i>Desenvolvimento</i>                        | <i>Testes</i>                           | <i>Testes / Produção</i>                         |



Verificação e Validação - V&V  
são termos comumente usados  
ao estudarmos teste e  
qualidade de software. Mas o  
que são exatamente?



As atividades de V&V tem por  
objetivo assegurar que o  
software seja adequado e  
atenda às necessidades, ou  
seja, a confirmação de que  
cumpra suas especificações.





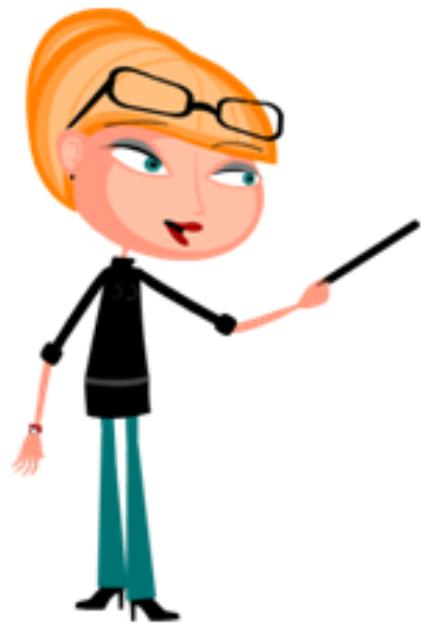
Verificação é a confirmação de que o software cumpre com suas especificações.  
Busca responder a pergunta:  
Estamos construindo corretamente o produto?



Enquanto validação é a confirmação de que o software está de acordo com o que o usuário deseja.  
Estamos construindo o produto certo?



As atividades de V&V podem ser **estáticas** (quando não exigem execução do software testado) e **dinâmicas** (obrigam a existência e execução do software testado)



#### Exemplos de atividades estáticas:

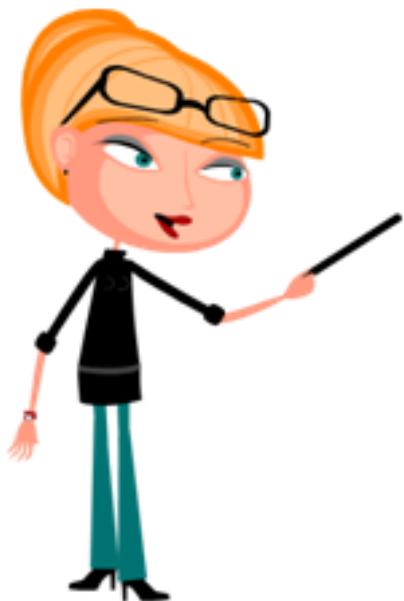
- Revisões de requisitos e modelos
- Inspeção de código
- walkthroughs

#### Exemplos de atividades dinâmicas:

- Teste unitário
- Teste de integração
- Teste de sistemas
- Teste de aceitação



As atividades de V&V podem ser **estáticas** (quando não exigem execução do software testado) e **dinâmicas** (obrigam a existência e execução do software testado)



#### Exemplos de atividades estáticas:

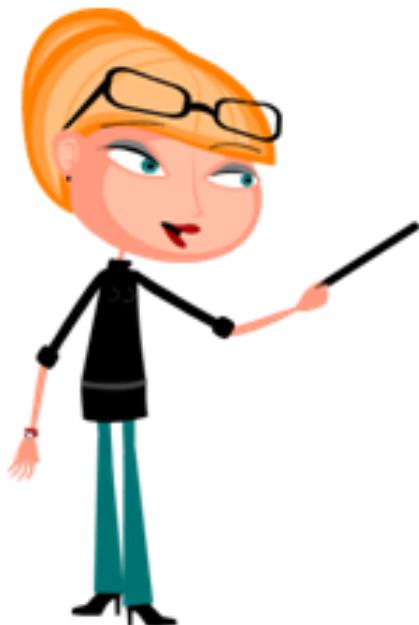
- Revisões de requisitos e modelos
- Inspeção de código
- walkthroughs

#### Exemplos de atividades dinâmicas:

- Teste unitário
- Teste de integração
- Teste de sistemas
- Teste de aceitação



As atividades de V&V podem ser **estáticas** (quando não exigem execução do software testado) e **dinâmicas** (obrigam a existência e execução do software testado)



#### Exemplos de atividades estáticas:

- Revisões de requisitos e modelos
- Inspeção de código
- walkthroughs

#### Exemplos de atividades dinâmicas:

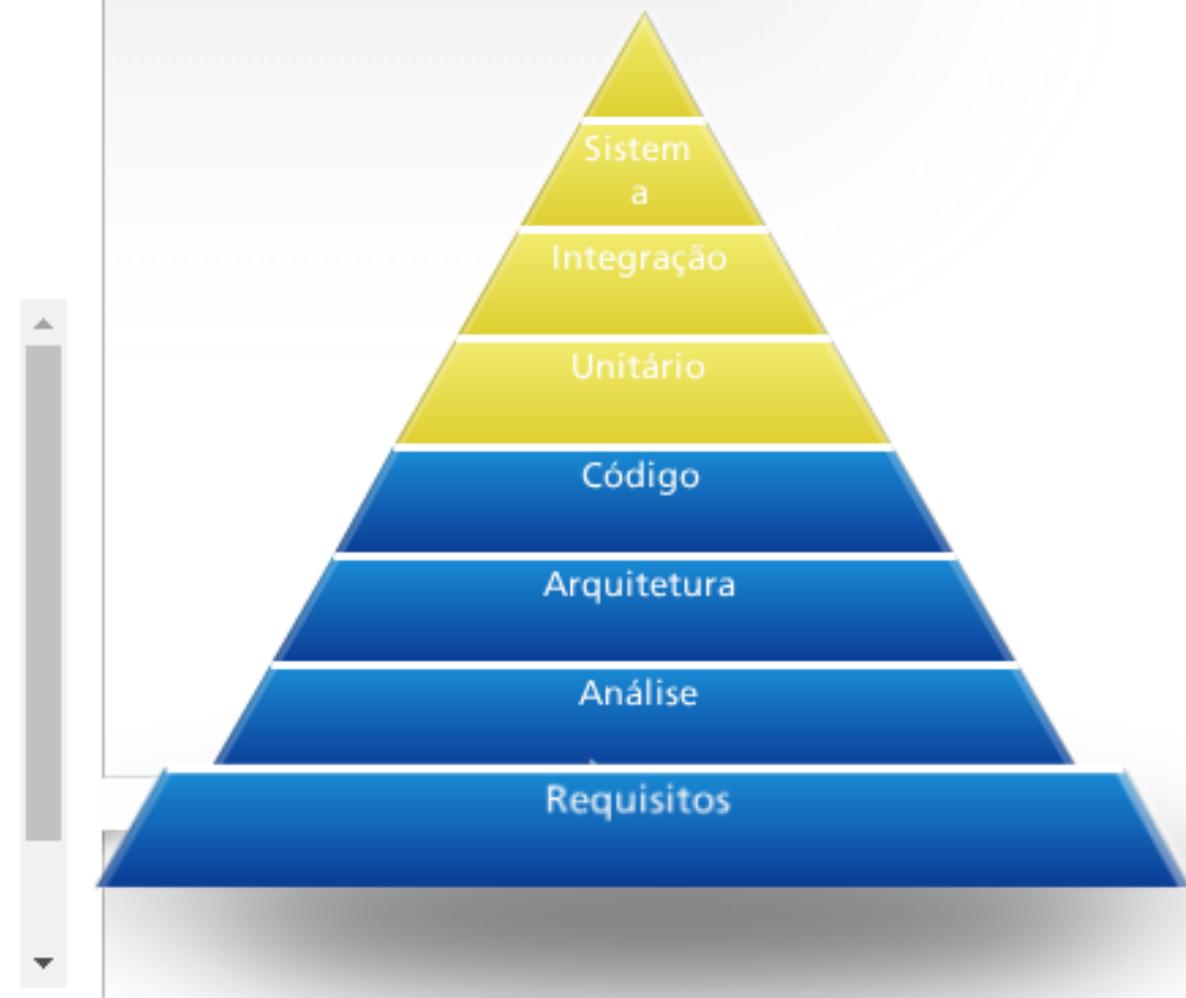
- Teste unitário
- Teste de integração
- Teste de sistemas
- Teste de aceitação

# Requisitos

A primeira etapa do desenvolvimento de software envolve o levantamento, análise e classificação de requisitos.

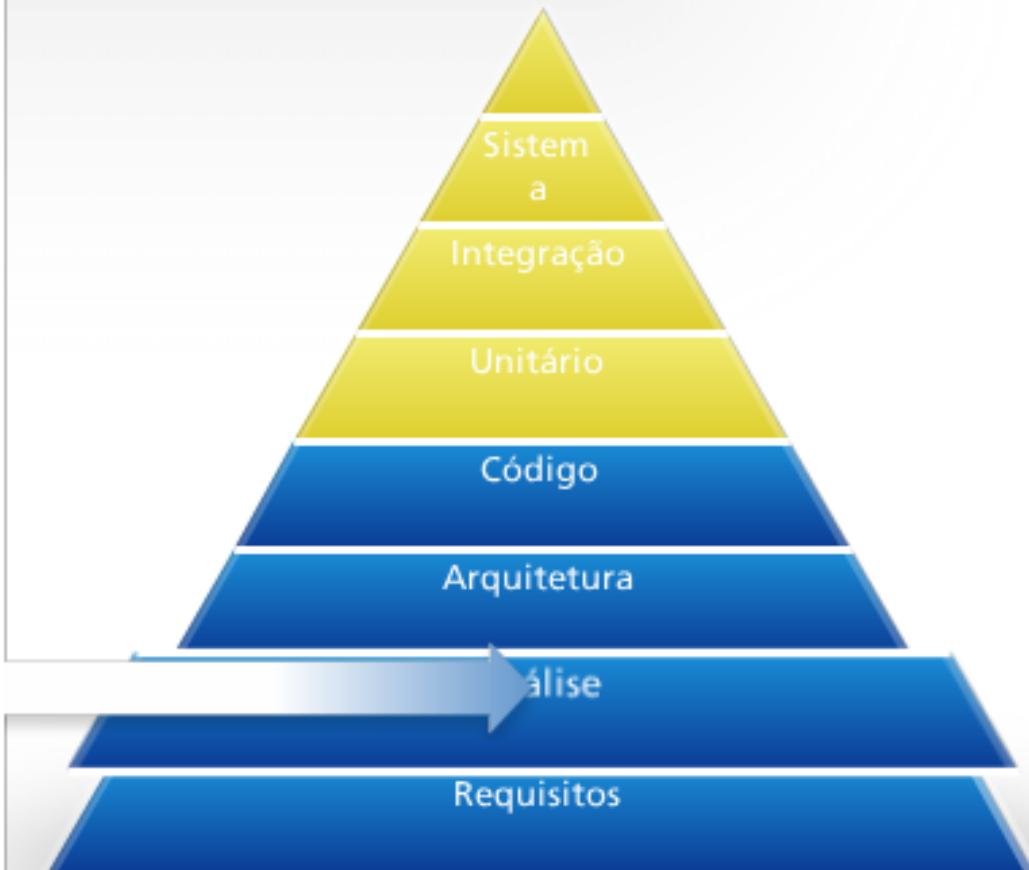
São os requisitos que irão determinar as especificações do sistema a ser desenvolvido. Desta forma, são também importantes para o processo de teste.

Atividades de revisão podem melhorar a qualidade dos requisitos, que serão base



# Análise

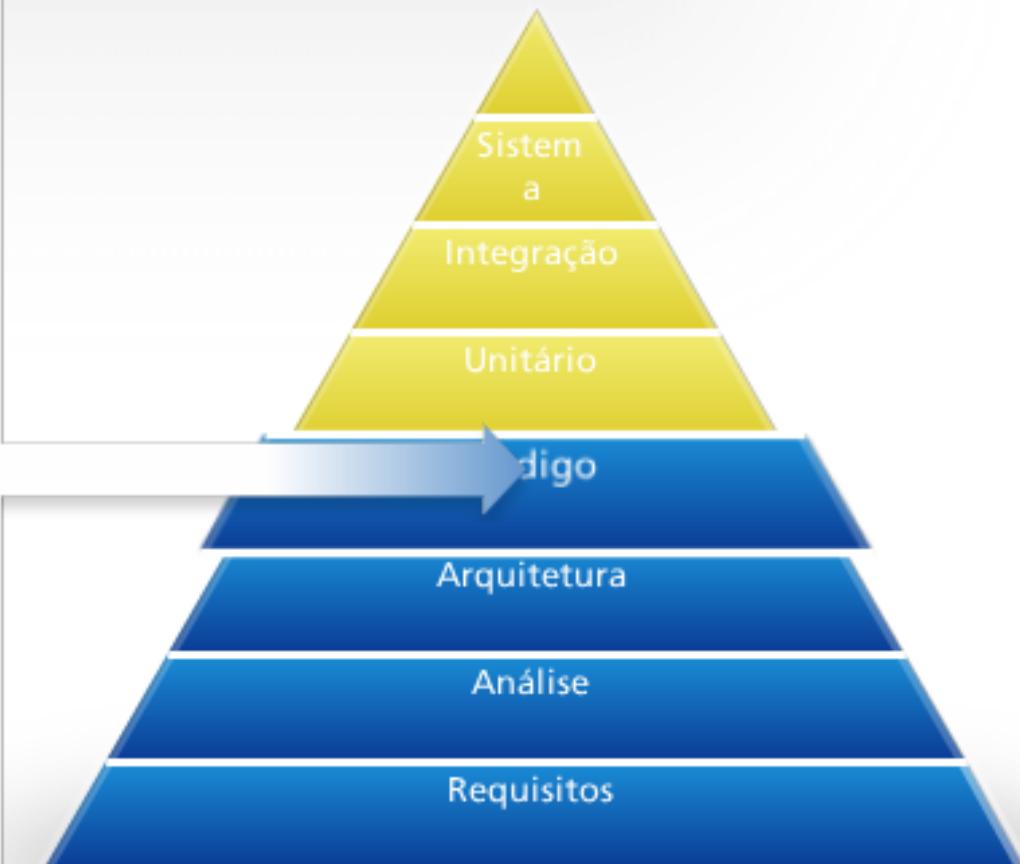
A análise dos artefatos gerados no processo de desenvolvimento, durante a etapa modelagem, como diagramas UML, documentos com detalhamento e classificação de requisitos, etc... pode apontar inconsistências ou erros de interpretação.



# Código

A técnica mais comum é a revisão de código, que pode ser feita de duas formas:

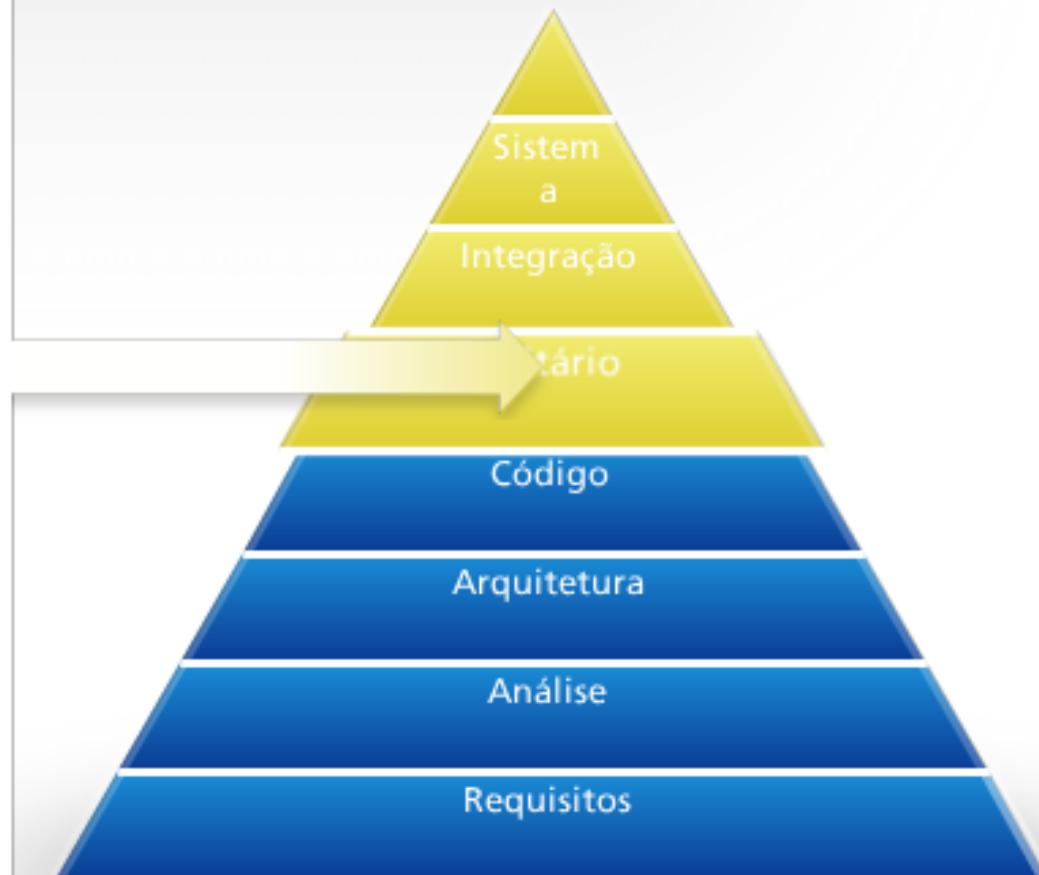
- 1. Walkthroughs:** o código fonte e a documentação correspondente são comparados pela equipe de revisão na busca por divergências.
- 2. Inspeção de código:** É mais formal que o walkthroughs. Agora a comparação entre o código fonte e a documentação é norteada por uma lista de aspectos predeterminados.



# Unitário

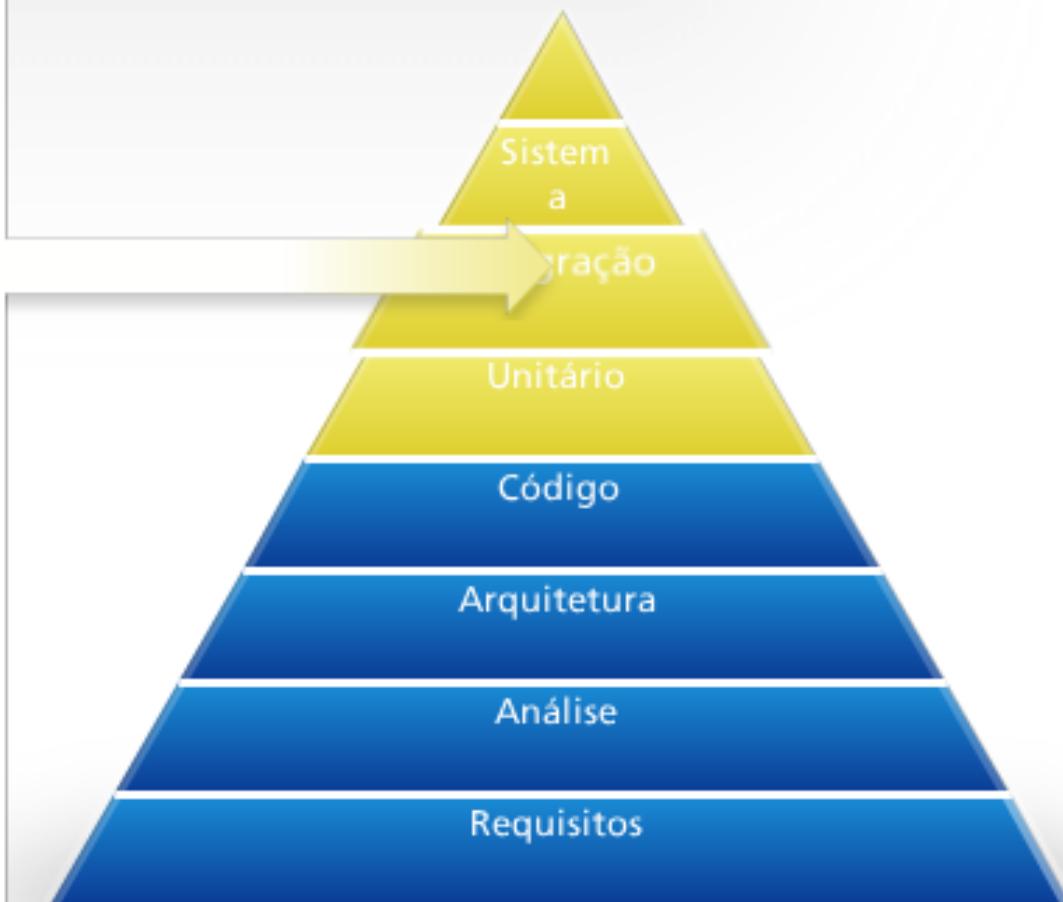
Testes unitários são implementações de teste feitas por desenvolvedores de software com intuito de testar o menor elemento testável, que geralmente em sistemas orientado a objetos são métodos.

Uma ferramenta conhecida entre os desenvolvedores Java, por exemplo, é o JUnit



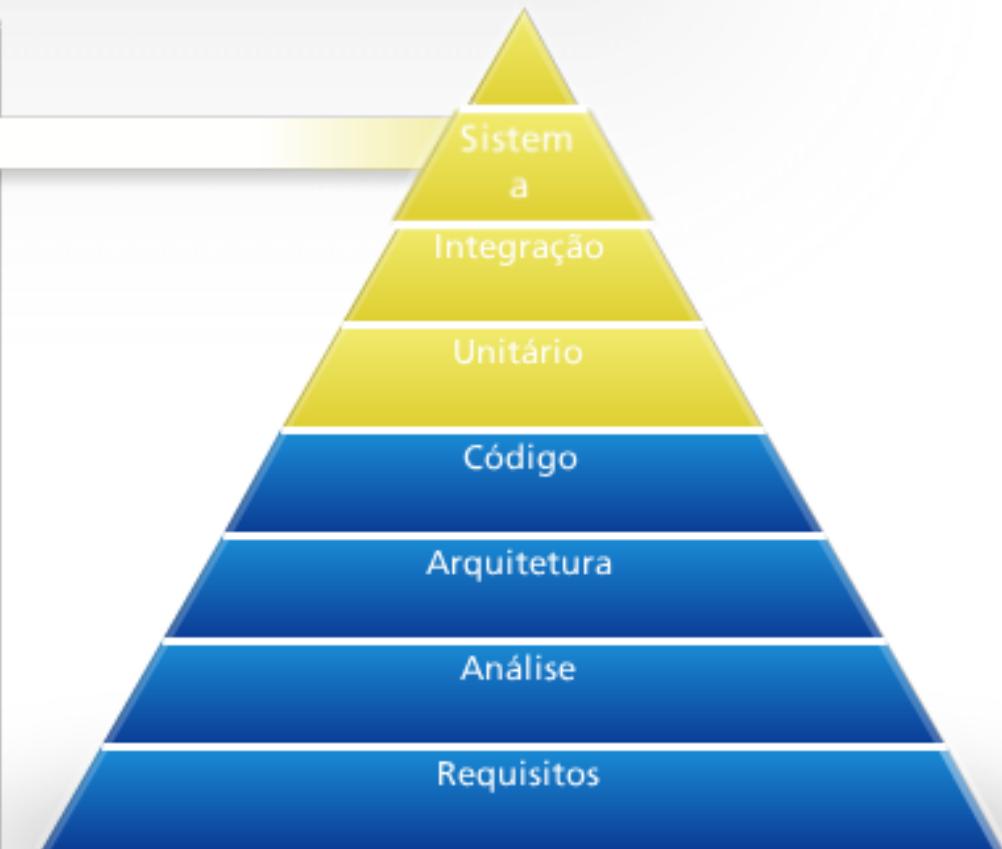
# Integração

Quando todos os componentes de um software foram testados, surge uma pergunta: quando estiverem integrados, continuarão funcionando? O teste de componentes individuais antecede o teste de integração. Contribui para assegurar a correção de componentes individuais, mas não é capaz de garantir que as dependências funcionais entre componentes estejam perfeitamente implementadas.



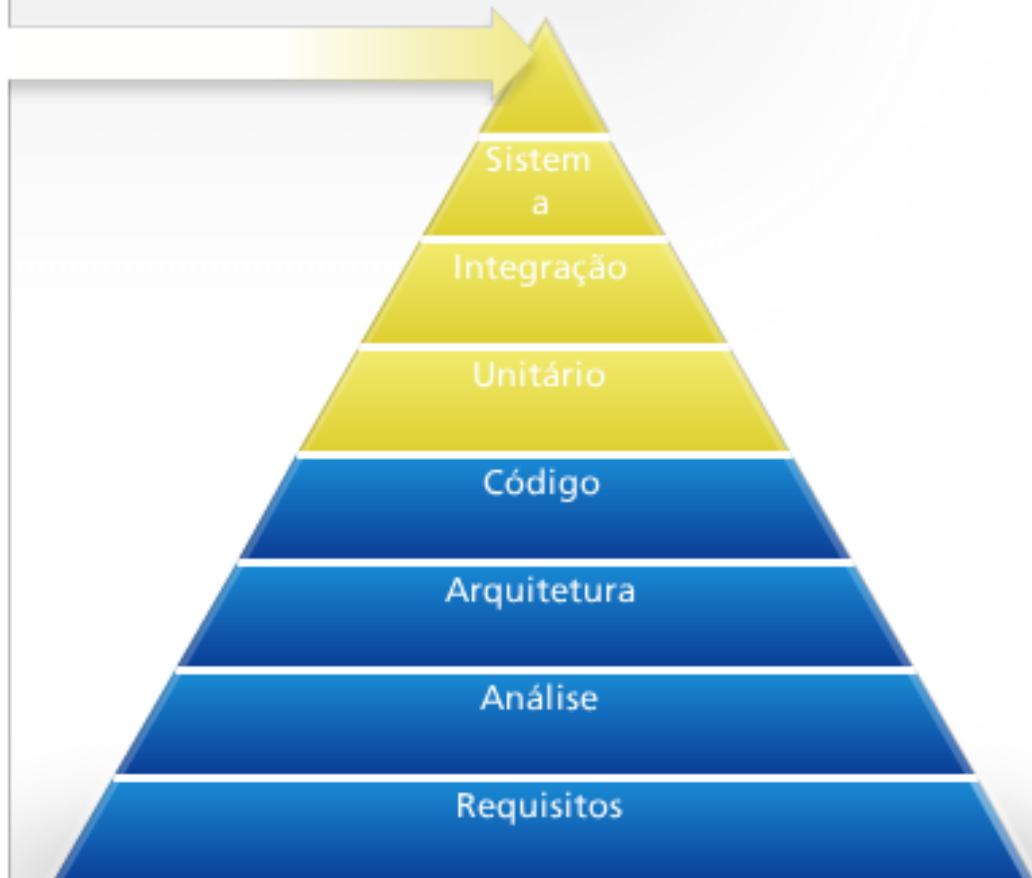
## Sistema

O teste de sistema é uma fase em que o sistema já está completamente integrado e é testado como um todo. Suas especificações (requisitos funcionais, não funcionais, regras de negócio e casos de uso) são exercitadas em um ambiente controlado.



# Aceitação

O teste de aceitação é a última ação de teste antes da implantação do software. A meta do teste de aceitação é verificar se o software está pronto e pode ser usado. O software é considerado pronto quando pode ser usado pelos usuários finais para executar as funções e as tarefas para as quais foi criado. A principal característica que distingue o teste de aceitação do teste de sistema é a participação (envolvimento) do usuário no teste.



# Técnicas de Teste de Software

## *Tipos de Teste de Software*

### ***Testes de Caixa-Branca (Estrutural)***

*Testes de Unidade*

*Teste de Integração*

### ***Testes de Caixa-Preta (Funcional)***

*Testes Funcionais*

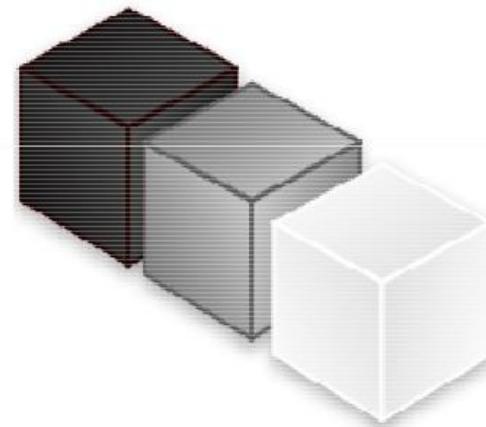
*Testes de Aceitação*

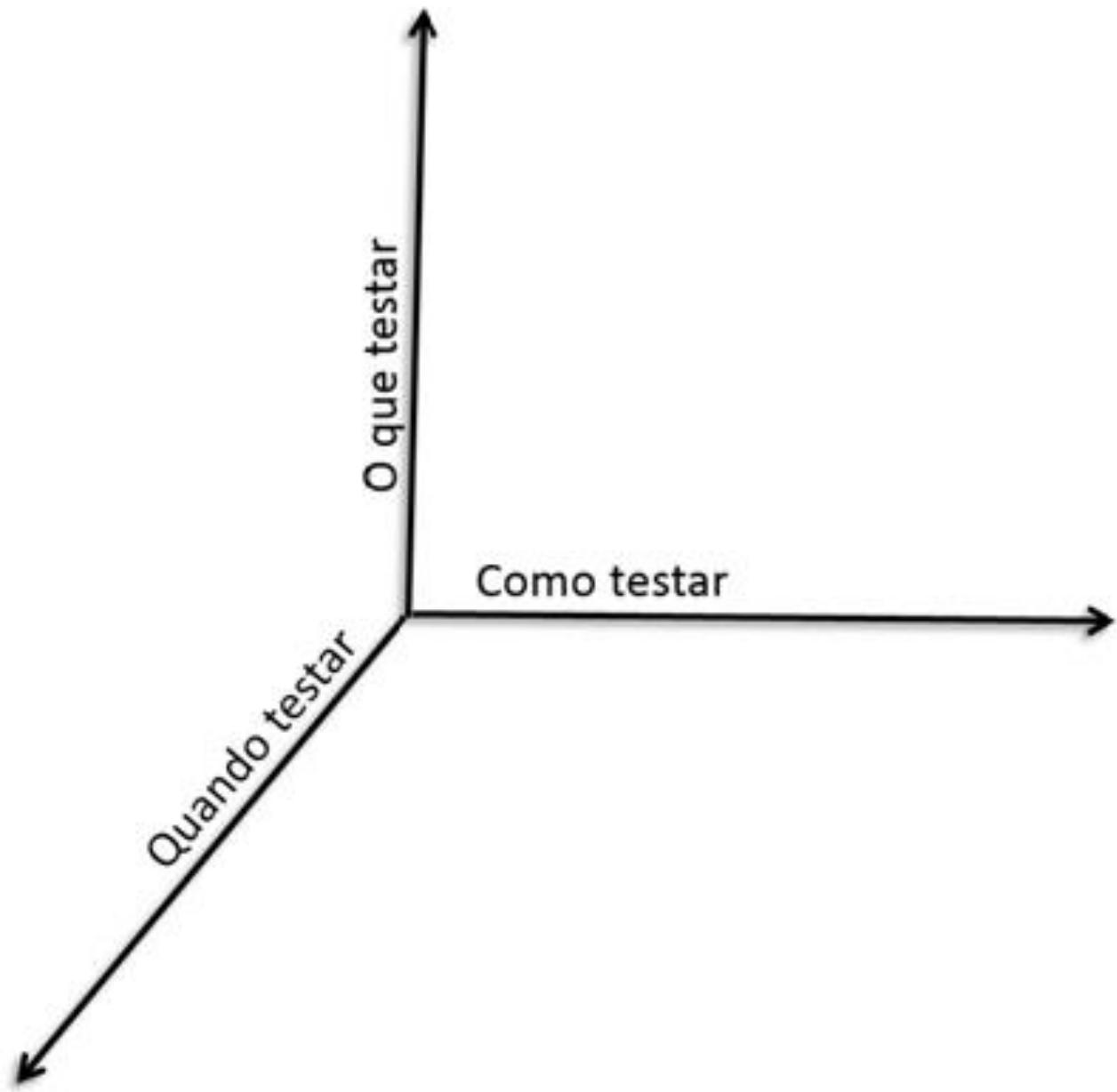
*Testes Exploratórios*

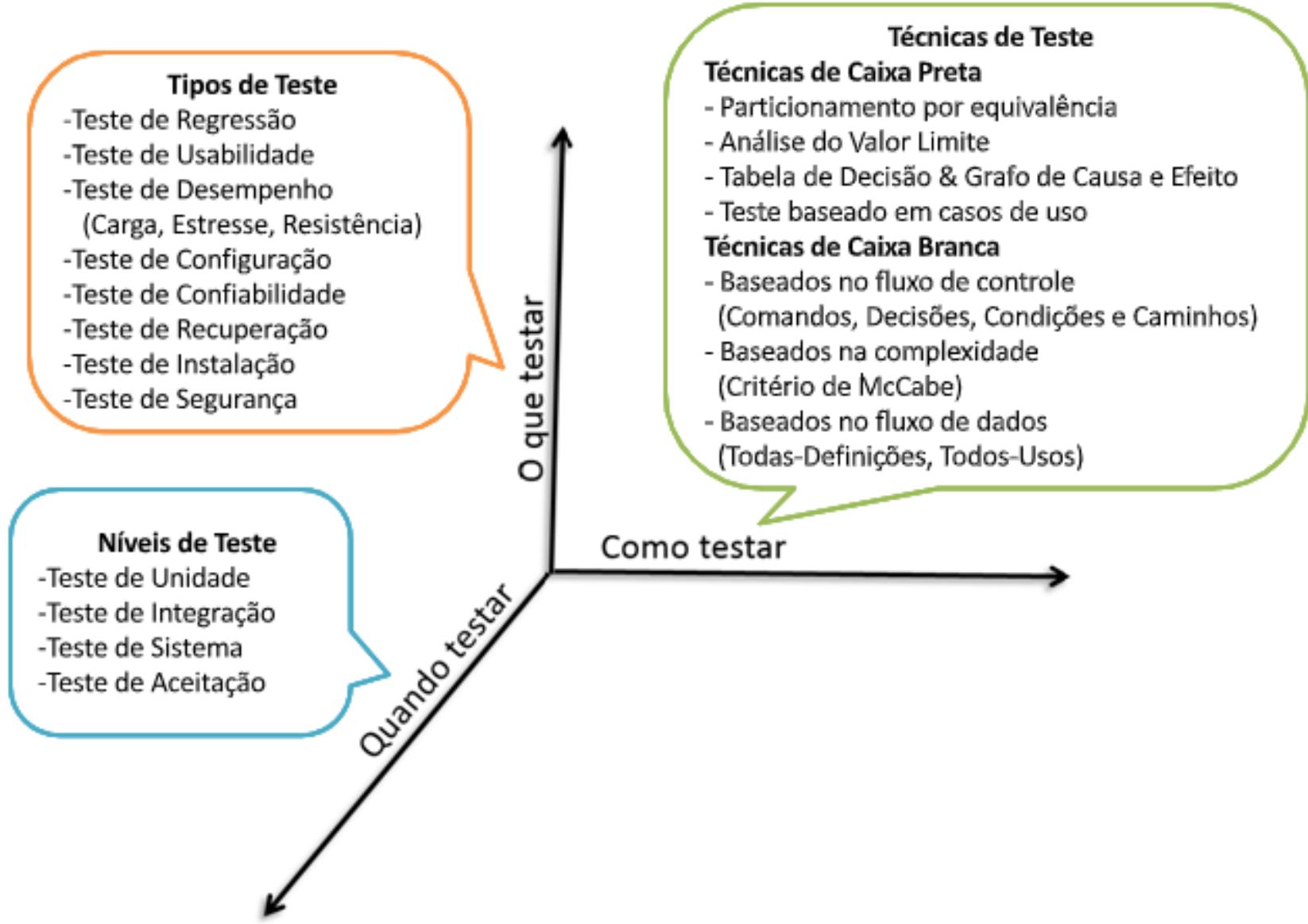
### ***Testes de Caixa-Cinza***

*Testes de Regressão*

*Testes de Cobertura*







- Tipos de Teste**
- Teste de Regressão
  - Teste de Usabilidade
  - Teste de Desempenho  
(Carga, Estresse, Resistência)
  - Teste de Configuração
  - Teste de Confiabilidade
  - Teste de Recuperação
  - Teste de Instalação
  - Teste de Segurança

- Níveis de Teste**
- Teste de Unidade
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceitação

O que testar

Quando testar

Como testar

É por aqui que  
iremos começar!

### Técnicas de Teste

#### Técnicas de Caixa Preta

- Particionamento por equivalência
- Análise do Valor Limite
- Tabela de Decisão & Grafo de Causa e Efeito
- Teste baseado em casos de uso

#### Técnicas de Caixa Branca

- Baseados no fluxo de controle  
(Comandos, Decisões, Condições e Caminhos)
- Baseados na complexidade  
(Critério de McCabe)
- Baseados no fluxo de dados  
(Todas-Definições, Todos-Usos)



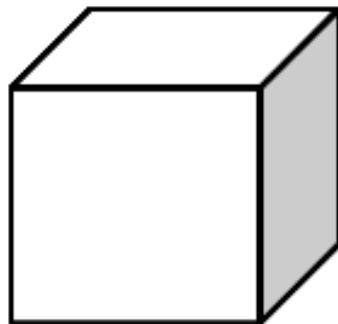


As técnicas de teste irão apresentar formas diferentes de elaboração de casos de teste explorando as diferentes características de um sistema!

Podemos dizer que as técnicas de teste definem a maneira como os testes são planejados e executados, ajudando na escolha do conjunto ideal de casos de teste!

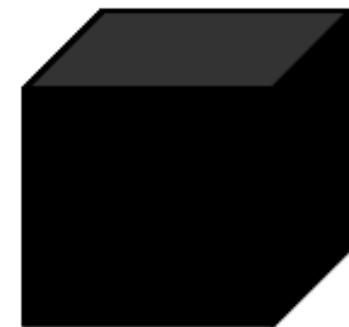


# Quais as técnicas de Teste Existentes?



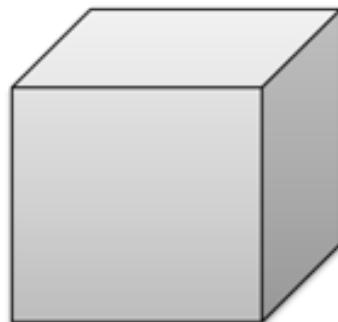
Caixa Branca

Também chamada caixa de vidro ou teste estrutural



Caixa Preta

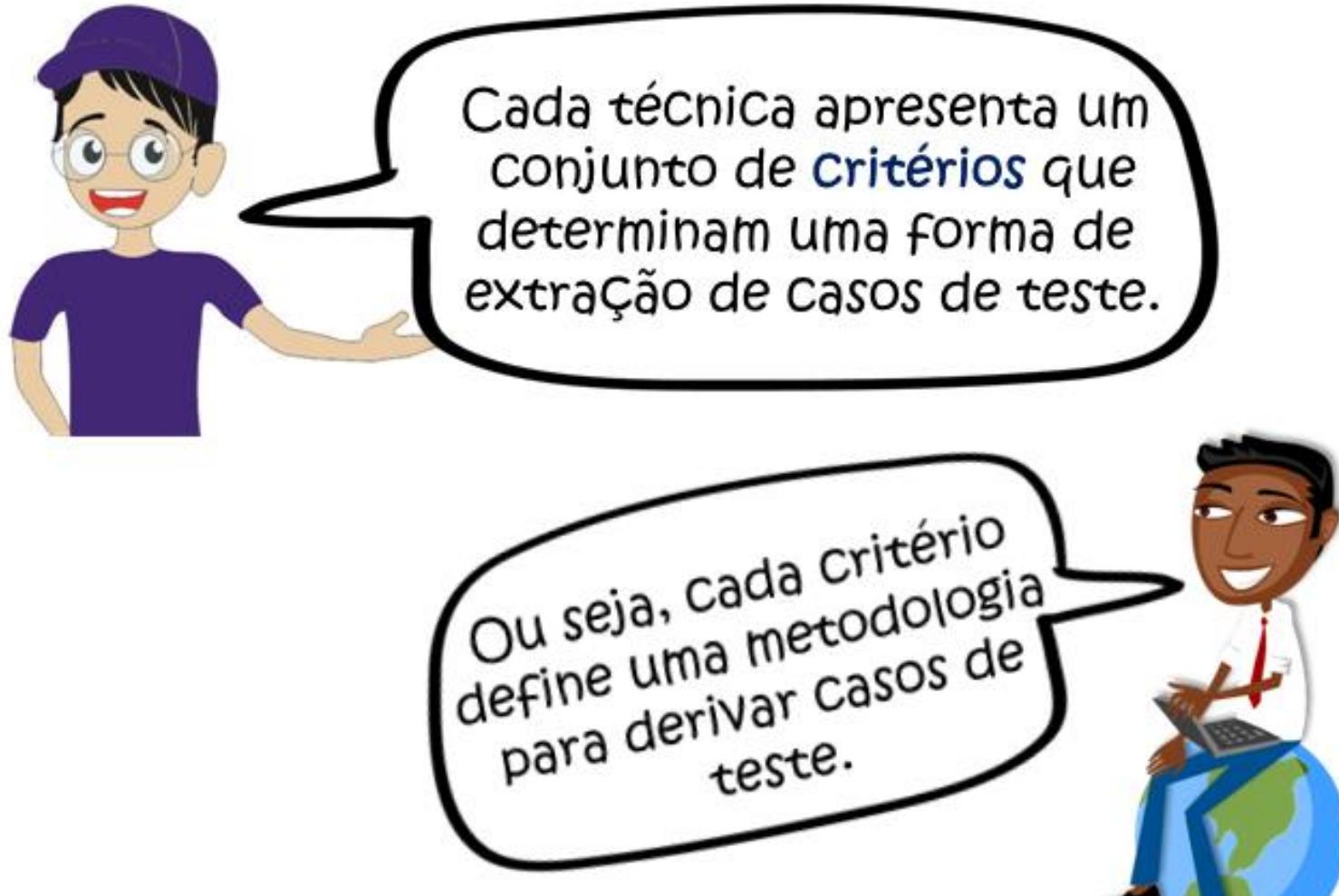
Também chamada teste funcional



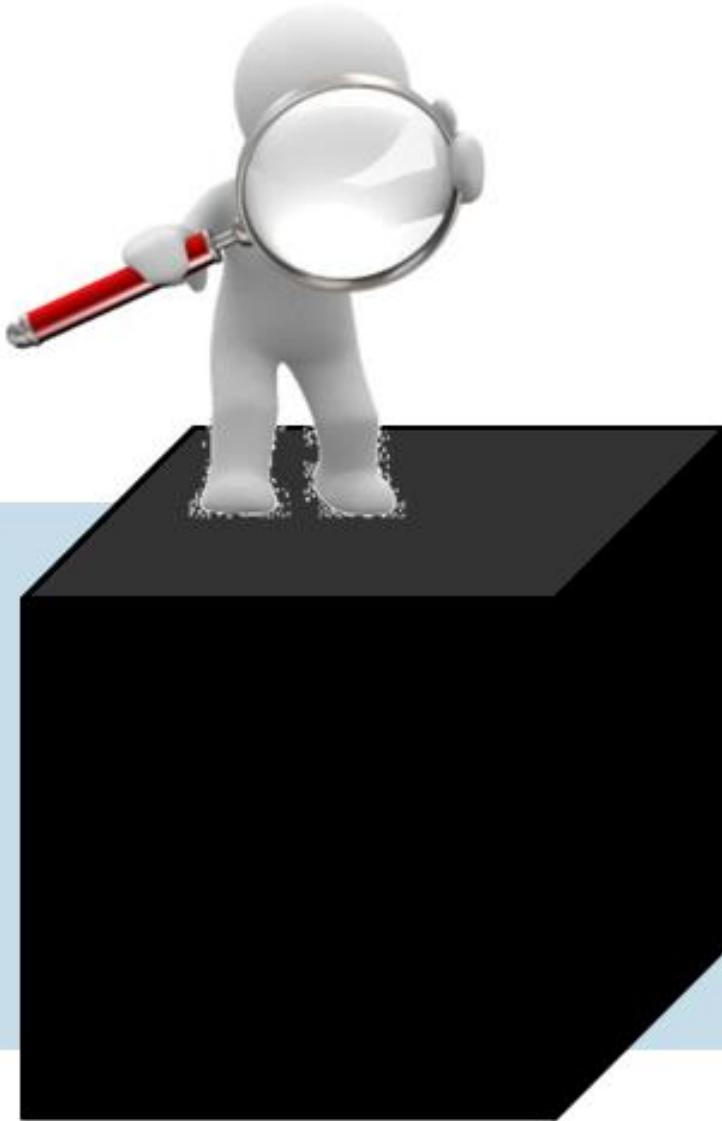
Caixa Cinza

Termo usado por alguns autores quando o projetista utiliza critérios das duas técnicas anteriores

# Critérios de Teste



# Técnica Caixa Preta



<https://www.youtube.com/embed/FeKCRxhLopg?autohide=0&showinfo=0&theme=light&rel=0>

# Conceitos – Técnica Caixa Preta



- Define se os requisitos desejados foram totalmente ou parcialmente satisfeitos pelo produto.
- Feito com base na funcionalidade do programa e avaliando se ele é capaz de gerar as saídas corretas para entradas de dados pré-estabelecidas.
- A estrutura do programa é ignorada para que o foco seja na funcionalidade. Não é necessário saber quais componentes do sistema são executados, e sim, o que ele deve fazer.

# Critérios – Técnica Caixa Preta

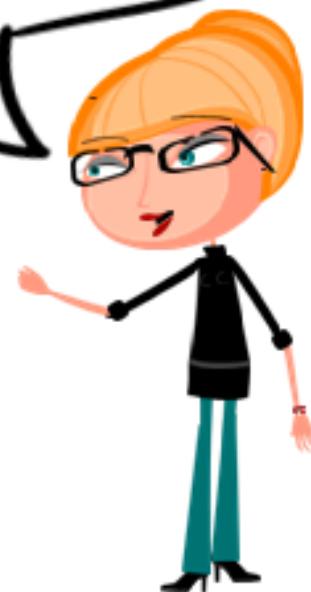
Partições de Equivalência

Análise do Valor Limite

Teste baseado em casos de uso

Grafo Causa-Efeito

Alguns critérios  
da técnica de  
caixa preta



# Exemplo prático – Observe as Funcionalidades

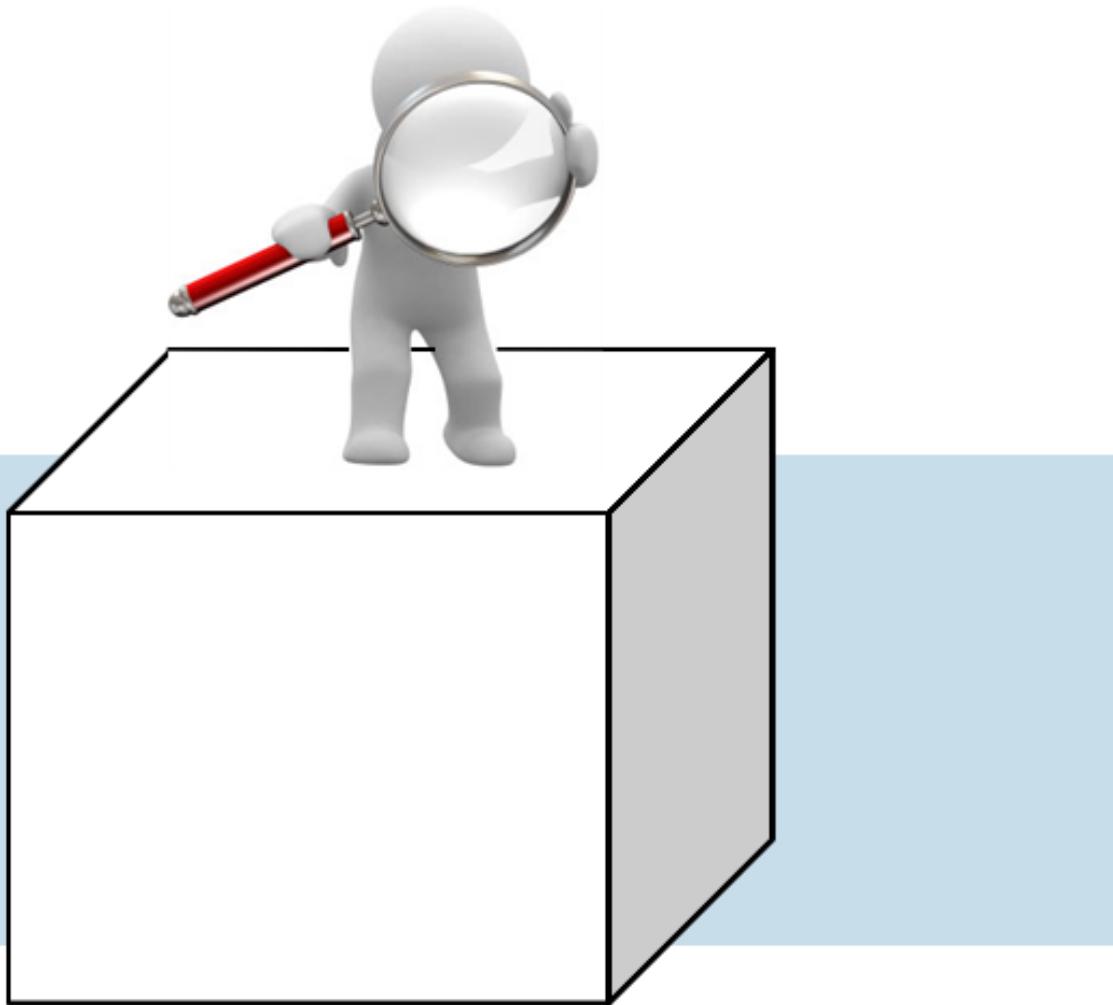
Como vimos, a técnica caixa preta consiste na aplicação de testes sem a necessidade do conhecimento do código fonte. Por exemplo, podemos testar o funcionamento de um aparelho de TV sem precisar saber como ele funciona por dentro.



Basta, para nosso exemplo, conhecermos os seguintes requisitos:

- O botão deve mudar apenas para próximo canal
- O botão deve mudar apenas para o canal anterior
- O botão deve apenas aumentar o volume
- O botão deve apenas diminuir o volume

# Técnica Caixa Branca





A técnica CAIXA BRANCA ou teste estrutural é responsável em avaliar o comportamento interno do componente de software.  
A função faz o que se espera dela?

Para essa técnica é necessário conhecer o código-fonte do programa.

```
public Usuario getUsuario(Integer codUsuario){  
    return getUsuarioPorCodigo(codUsuario);  
}
```

# Técnica Caixa Branca: Critérios Baseados no Fluxo de Controle



Utilizando o conhecimento detalhado do código-fonte é possível criar uma série de testes de modo a exercitar todos os seus elementos internos.

- Está retornando o usuário certo?

```
Public Usuario getUsuario(Integer codUsuario){  
    return getUsuarioPorCodigo(codUsuario);  
}
```

# Técnica Caixa Branca: Critérios Baseados no Fluxo de Dados



Dentre os critérios mais conhecidos da técnica caixa branca podemos citar:

Teste de Caminhos

Teste de Comandos

Teste de Decisões

# Teste de caminhos



# Exemplo

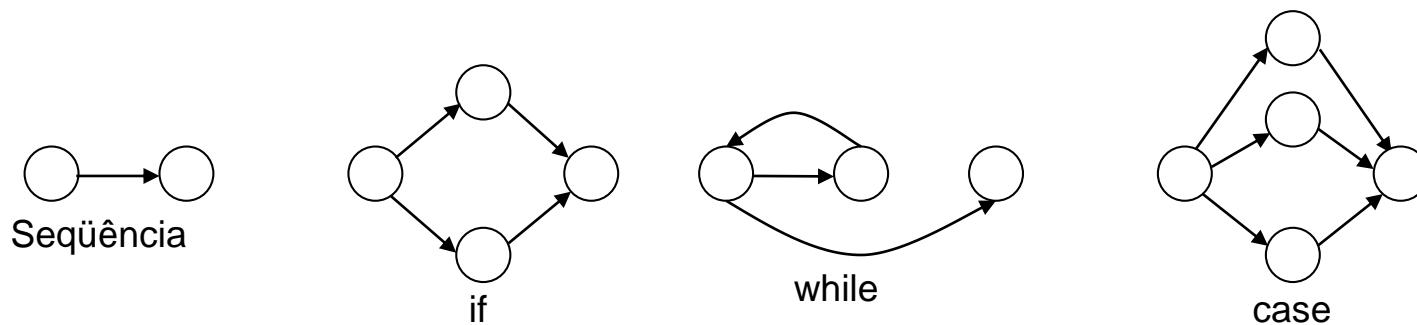


# Teste de caminho básico- exercício

- É uma **técnica de teste de caixa branca** que possibilita que o projetista do caso de teste derive uma medida de complexidade lógica de um projeto procedural e use essa medida como guia para definir **um conjunto básico de caminhos de execução**.

Notação de grafo de fluxo:

- *notação simples para representação do fluxo de controle, que descreve o fluxo lógico:*



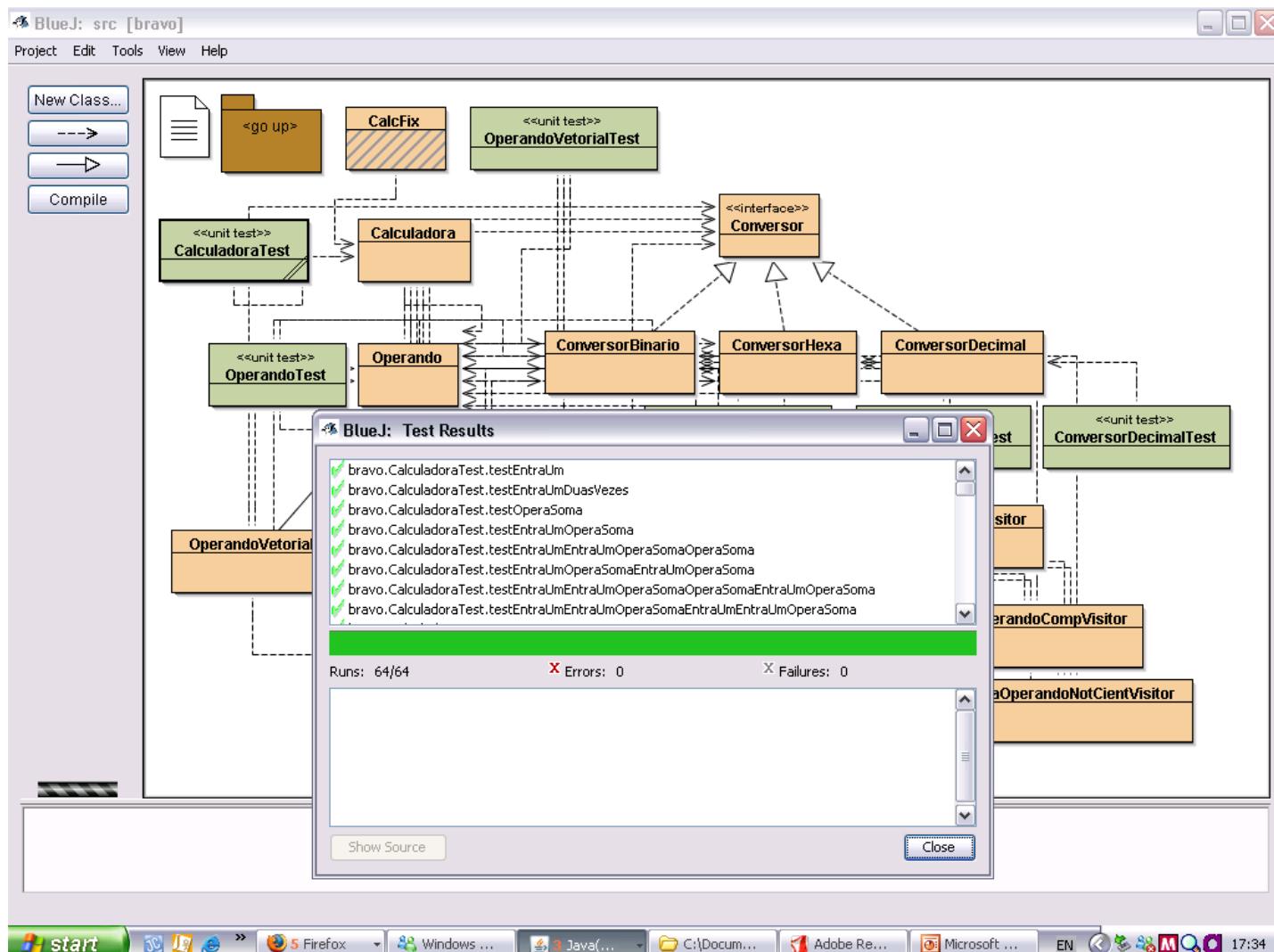
# Complexidade Ciclomática

- 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.

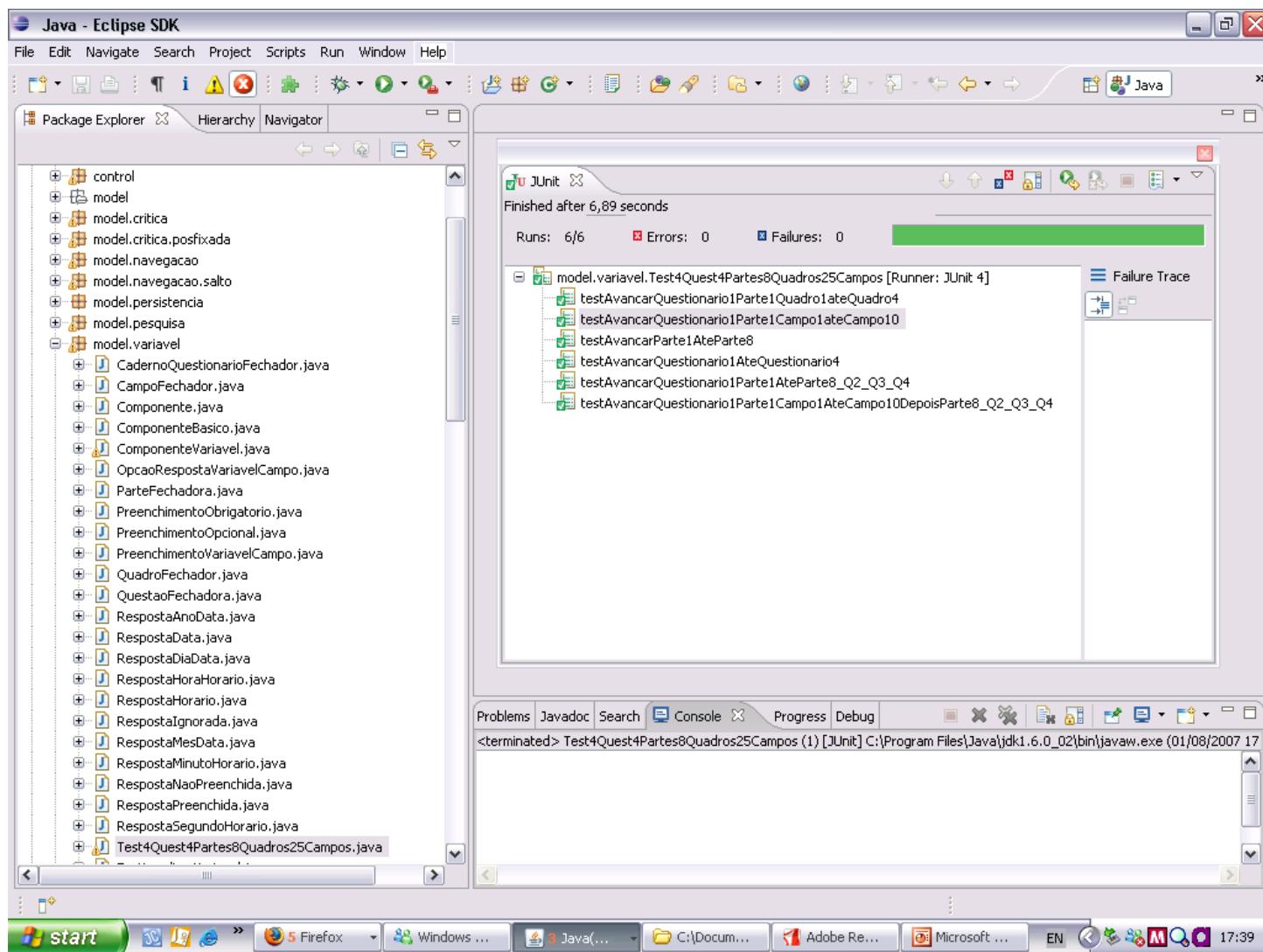
# Test-Driven Development (TDD)

- 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.*
- Técnicas
  - *"Fake It Til You Make It": faça um teste rodar simplesmente fazendo método retornar constante*
  - *Implementação óvia: se operações são simples, implemente-as e faça que os testes rodem*

# Plugin JUnit (BlueJ)



# Plugin JUnit (Eclipse)



# Ferramentas para Testes das GUI's

- Caso específico: resposta de servidores Web
  - ▣ *Verificar se uma página HTML ou XML contém determinado texto ou determinado elemento*
  - ▣ *Verificar se resposta está de acordo com dados passados na requisição: testes funcionais tipo "caixa-preta"*
- Soluções (extensões do JUnit)
  - ▣ *HttpUnit e ServletUnit:*
    - permite testar dados de árvore DOM HTML gerada
  - ▣ *JXWeb (combinação do JXUnit com HttpUnit)*
    - permite especificar os dados de teste em arquivos XML
    - arquivos de teste Java são gerados a partir do XML
  - ▣ *XMLUnit*
    - extensão simples para testar árvores XML
  - ▣ *Onde encontrar: ([httpunit/jxunit/xmlunit](http://httpunit/jxunit/xmlunit)).sourceforge.net*
- Outras: Cactus, JUnitPerf, JUnitEE...

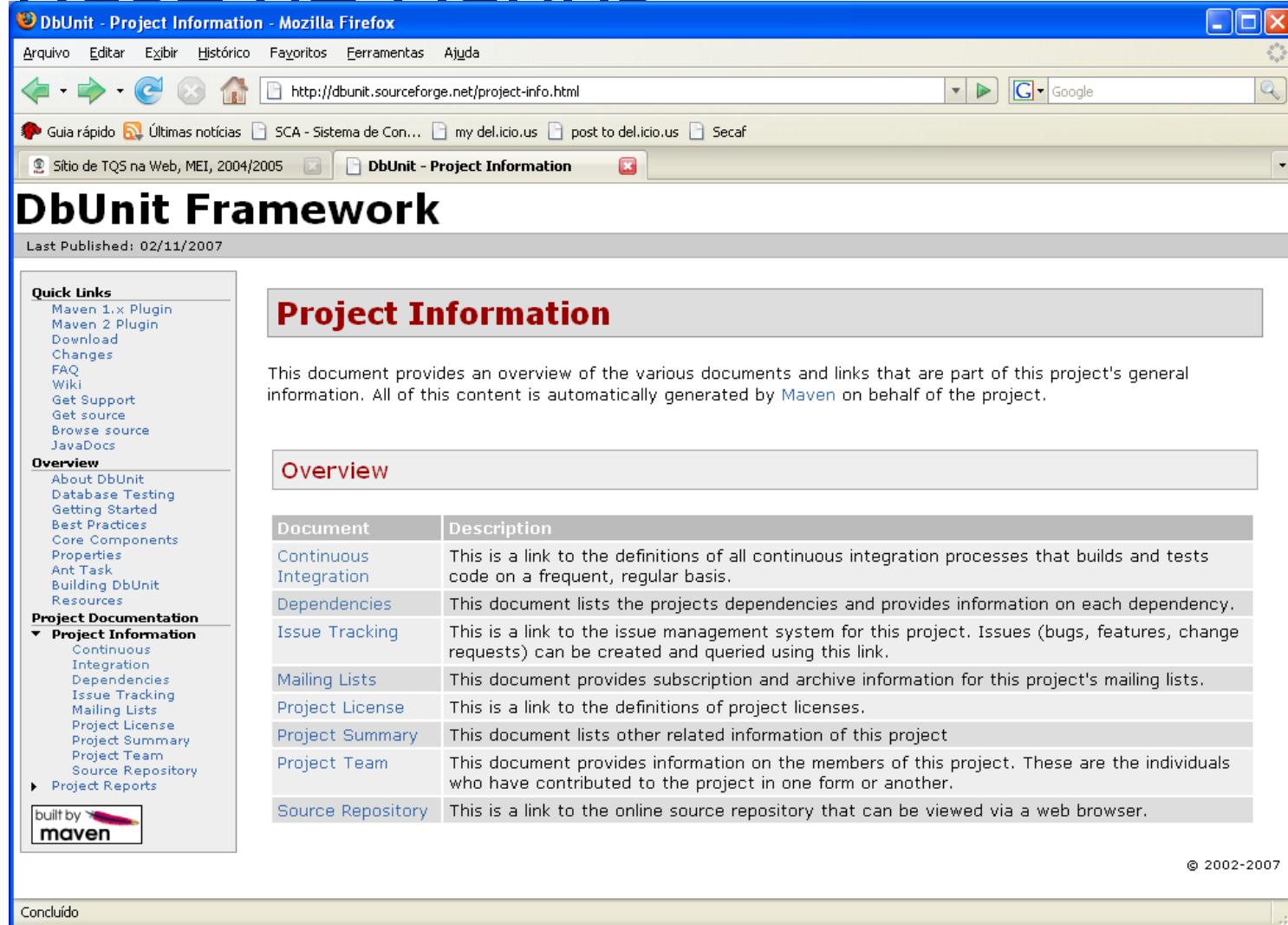
# Ferramenta para Testes de Performance

- JUnitPerf ([www.clarkware.com](http://www.clarkware.com))
  - ▣ Coleção de decoradores para medir performance e escalabilidade em testes JUnit existentes
- TimedTest
  - ▣ Executa um teste e mede o tempo transcorrido
  - ▣ Define um tempo máximo para a execução. Teste falha se execução durar mais que o tempo estabelecido
- LoadTest
  - ▣ Executa um teste com uma carga simulada
  - ▣ Utiliza timers para distribuir as cargas usando distribuições randômicas
  - ▣ Combinado com TimerTest para medir tempo com carga
- ThreadedTest
  - ▣ Executa o teste em um thread separado

# Frameworks para Testes de Unidade

- Similares ao JUnit (linguagem Java):
  - *Python*
    - PyUnit
  - *C++*
    - CppUnit
  - *Perl*
    - PerlUnit
  - *.NET*
    - NUnit, NUnitForms, dotUnit, EasyMock.NET, csUnit

# Testes envolvendo acesso a Base de Dados



The screenshot shows a Mozilla Firefox browser window with the title "DbUnit - Project Information - Mozilla Firefox". The address bar displays the URL <http://dbunit.sourceforge.net/project-info.html>. The page content is the "Project Information" section of the DbUnit framework's website.

**Project Information**

This document provides an overview of the various documents and links that are part of this project's general information. All of this content is automatically generated by [Maven](#) on behalf of the project.

**Overview**

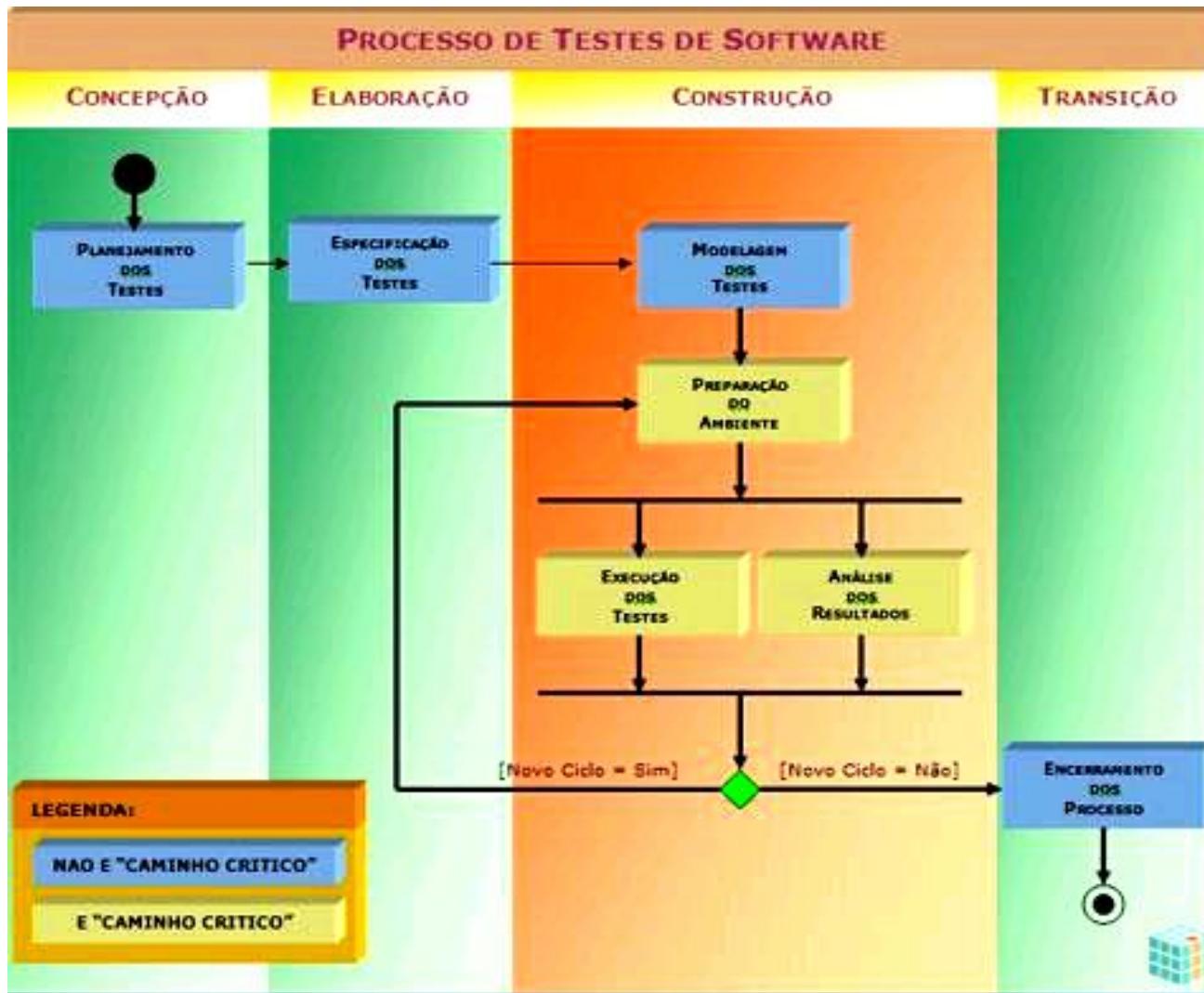
| Document               | Description                                                                                                                                              |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Continuous Integration | This is a link to the definitions of all continuous integration processes that builds and tests code on a frequent, regular basis.                       |
| Dependencies           | This document lists the projects dependencies and provides information on each dependency.                                                               |
| Issue Tracking         | This is a link to the issue management system for this project. Issues (bugs, features, change requests) can be created and queried using this link.     |
| Mailing Lists          | This document provides subscription and archive information for this project's mailing lists.                                                            |
| Project License        | This is a link to the definitions of project licenses.                                                                                                   |
| Project Summary        | This document lists other related information of this project                                                                                            |
| Project Team           | This document provides information on the members of this project. These are the individuals who have contributed to the project in one form or another. |
| Source Repository      | This is a link to the online source repository that can be viewed via a web browser.                                                                     |

built by  maven

Concluído

© 2002-2007

# Processo de Teste de Software na visão do RUP



# Planejamento de Testes

- 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

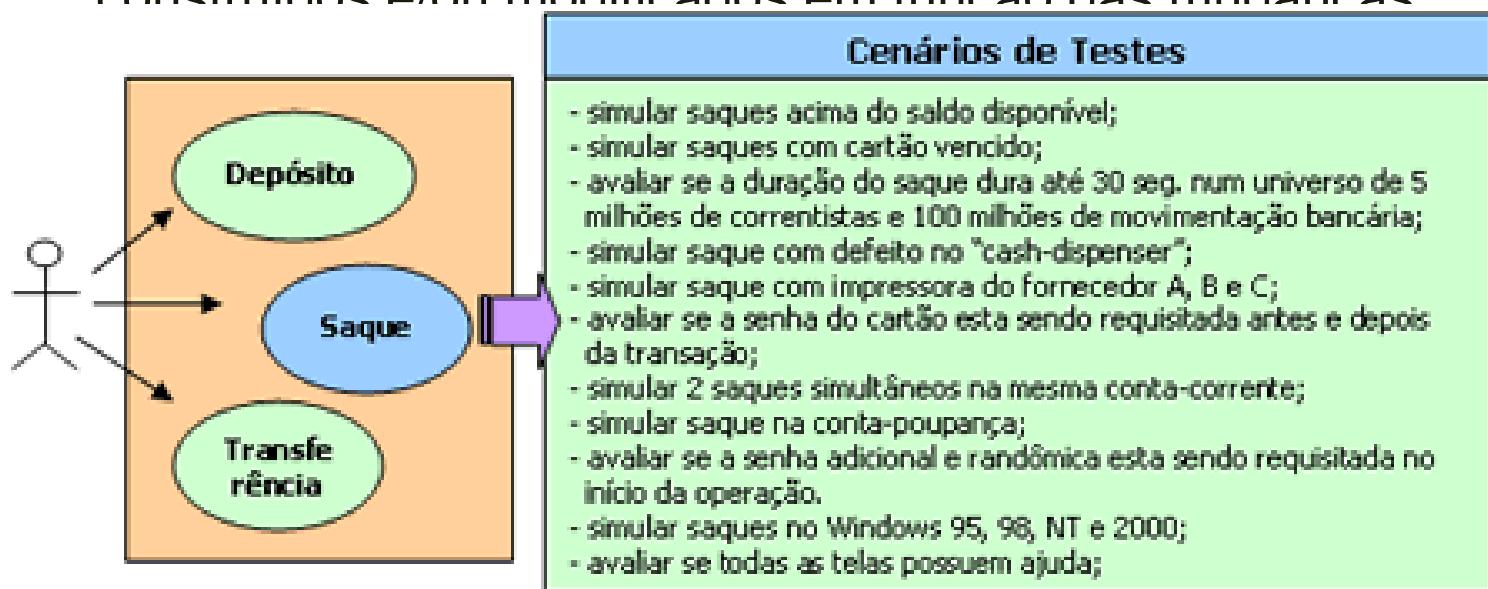


Figura - Levantamento dos Cenários sem aplicar os conceitos de Categorização

# Especificação dos Testes (Categorias)

| Funcional                                                                                                                                                                                                                                                                                                                                              | Segurança                                                                                                                                                                                                                                                                                                                                              | Usabilidade                                                                                                                                                                                                                                                                                          | Performance                                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>- simular saques acima do saldo disponível;</li> <li>- simular saque na conta-poupança;</li> <li>- <b>simular saque acima do valor do limite da conta;</b></li> <li>- <b>simular saque com valores não múltiplos das notas;</b></li> <li>- <b>simular saque com valores não múltiplos das notas;</b></li> </ul> | <ul style="list-style-type: none"> <li>- simular saques com cartão vencido;</li> <li>- avaliar se a senha do cartão está sendo requisitada antes e depois da transação;</li> <li>- avaliar se a senha adicional e randômica está sendo requisitada no início da operação;</li> <li>- <b>simular saque noturno acima do valor permitido;</b></li> </ul> | <ul style="list-style-type: none"> <li>- avaliar se todas as telas possuem ajuda;</li> <li>- <b>avaliar se mensagens são claras e objetivas;</b></li> <li>- avaliar se o padrão visual é mantido em todos os momentos;</li> <li>- avaliar se todas as operações possuem caminhos de fuga;</li> </ul> | <ul style="list-style-type: none"> <li>- avaliar se a duração do saque dura até 30 seg. num universo de 5 milhões de correntistas e 100 milhões de movimentação bancária;</li> <li>- garantir que manipulação com dispositivos físicos no saque não ultrapassem 10 seg. da operação;</li> </ul> |
| Carga e Concorrência                                                                                                                                                                                                                                                                                                                                   | Configuração                                                                                                                                                                                                                                                                                                                                           | Recuperação                                                                                                                                                                                                                                                                                          | Contingência                                                                                                                                                                                                                                                                                    |
| <ul style="list-style-type: none"> <li>- simular 2 saques simultâneos na mesma conta-corrente;</li> <li>- <b>simular 10.000 saques simultâneos;</b></li> </ul>                                                                                                                                                                                         | <ul style="list-style-type: none"> <li>- simular saque com impressora do fornecedor A, B e C;</li> <li>- simular saques no Windows 95, 98, NT e 2000;</li> <li>- <b>simular saque com impressora do fornecedor X, Y e Z;</b></li> </ul>                                                                                                                | <ul style="list-style-type: none"> <li>- simular saque com defeito no "cash-dispenser";</li> <li>- <b>simular saque com defeito na impressora;</b></li> <li>- simular saque com falha de conexão com a central;</li> <li>- simular saque com queda de energia;</li> </ul>                            | <ul style="list-style-type: none"> <li>- disparar processo de instalação emergencial;</li> </ul>                                                                                                                                                                                                |

# 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).

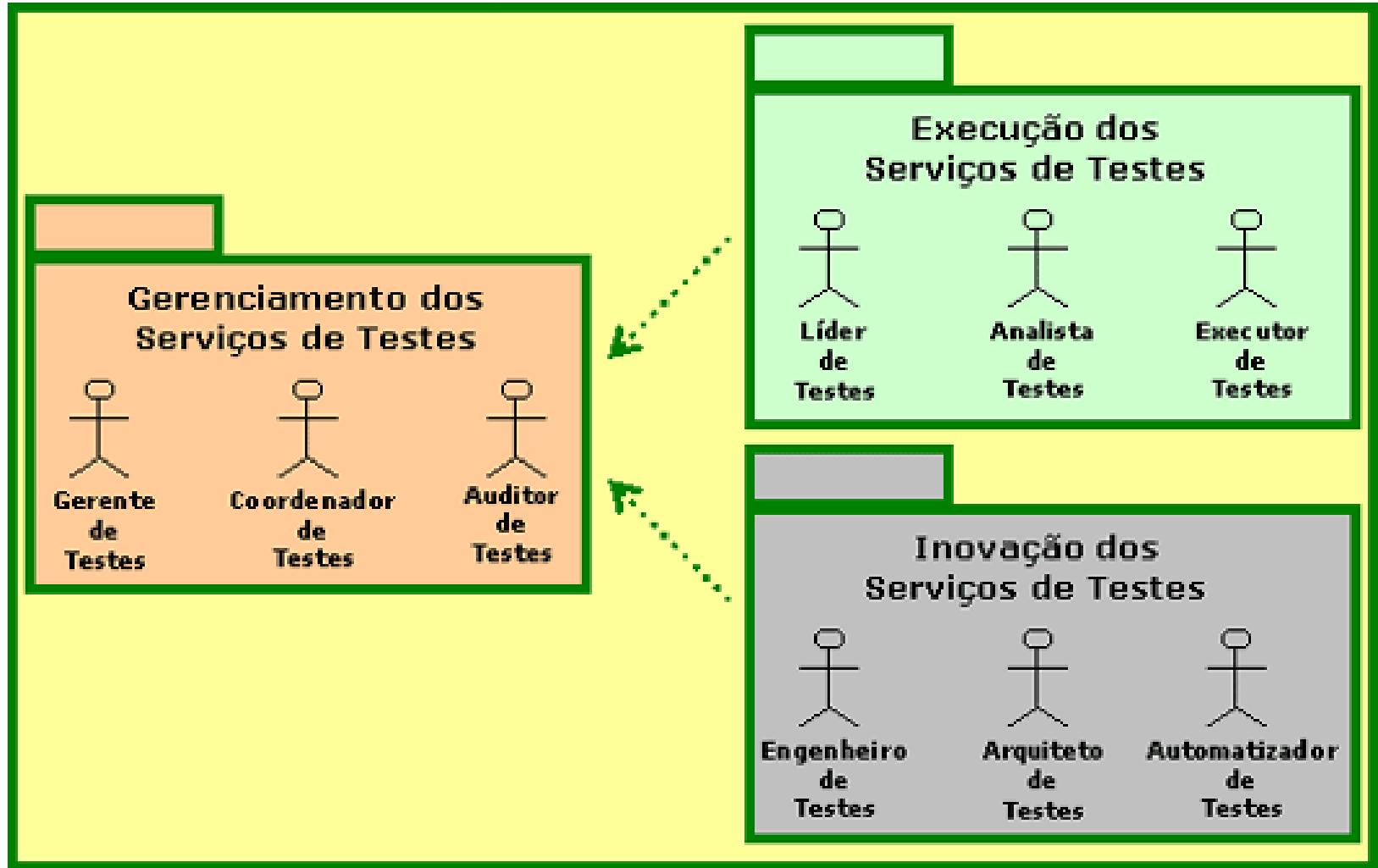
# Execução dos Testes

- 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" reconfirmedo.

# Equipes de Teste



## Norma IEEE 829-1998

- A norma IEEE 829-1998 descreve um conjunto de documentos para as atividades de teste de um produto de software. Os documentos cobrem as tarefas de planejamento, especificação e relato de testes.