



# BandTec

DIGITAL SCHOOL

# Recados Paroquiais



## Bibliografia:

Engenharia de Software 8º Edição / Ian Sommerville

Engenharia de Software 6º Edição / Roger Pressman



## Adicional

- Code Complete
- SWEBOK
- entre outros

# Nosso Objetivo

Aprender/Ensinar processos, métodos e ferramentas para construção e manutenção de softwares profissionais.

PRAGMATISMO

# Objetivo da Aula

- Qualidade



# Engenharia de Software

Qualidade de Software..Continuando

Professor Esp. Gerson Santos

# Antes de começar... Vamos alinhar

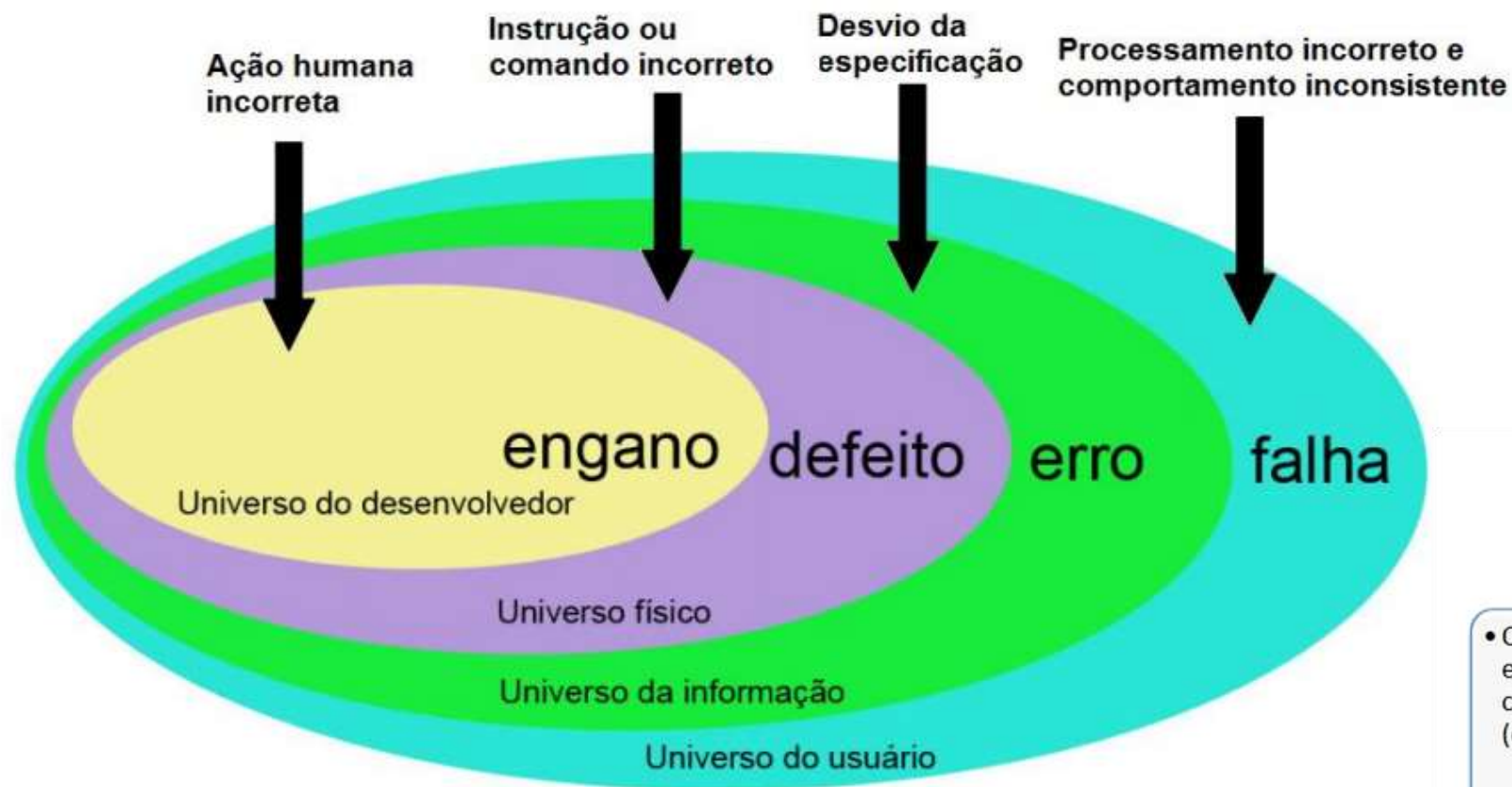


Figura 1 - Engano x defeito x erro x falha  
Fonte: Adaptado de Barbosa et al., 2000.



# Antes de começar... Vamos alinhar

Não vamos entrar na discussão do que é BUG, defeito, erro, etc.

O livro mais simples é o do Pressman. Vamos por ele:

BUG : Antes de liberar o software

DEFEITO: Depois de liberar o software

# Verificação & Validação..mais divergências

A Verificação refere-se ao conjunto de tarefas que garantem que o software implemente corretamente uma função específica.

Validação refere-se a um conjunto diferente de tarefas que asseguram que o software que foi construído seja rastreável aos requisitos do cliente.

Boehm [Boe81] afirma isso de outra maneira:

Verificação: "Estamos construindo o produto corretamente?"

Validação: "Estamos criando o produto certo?"

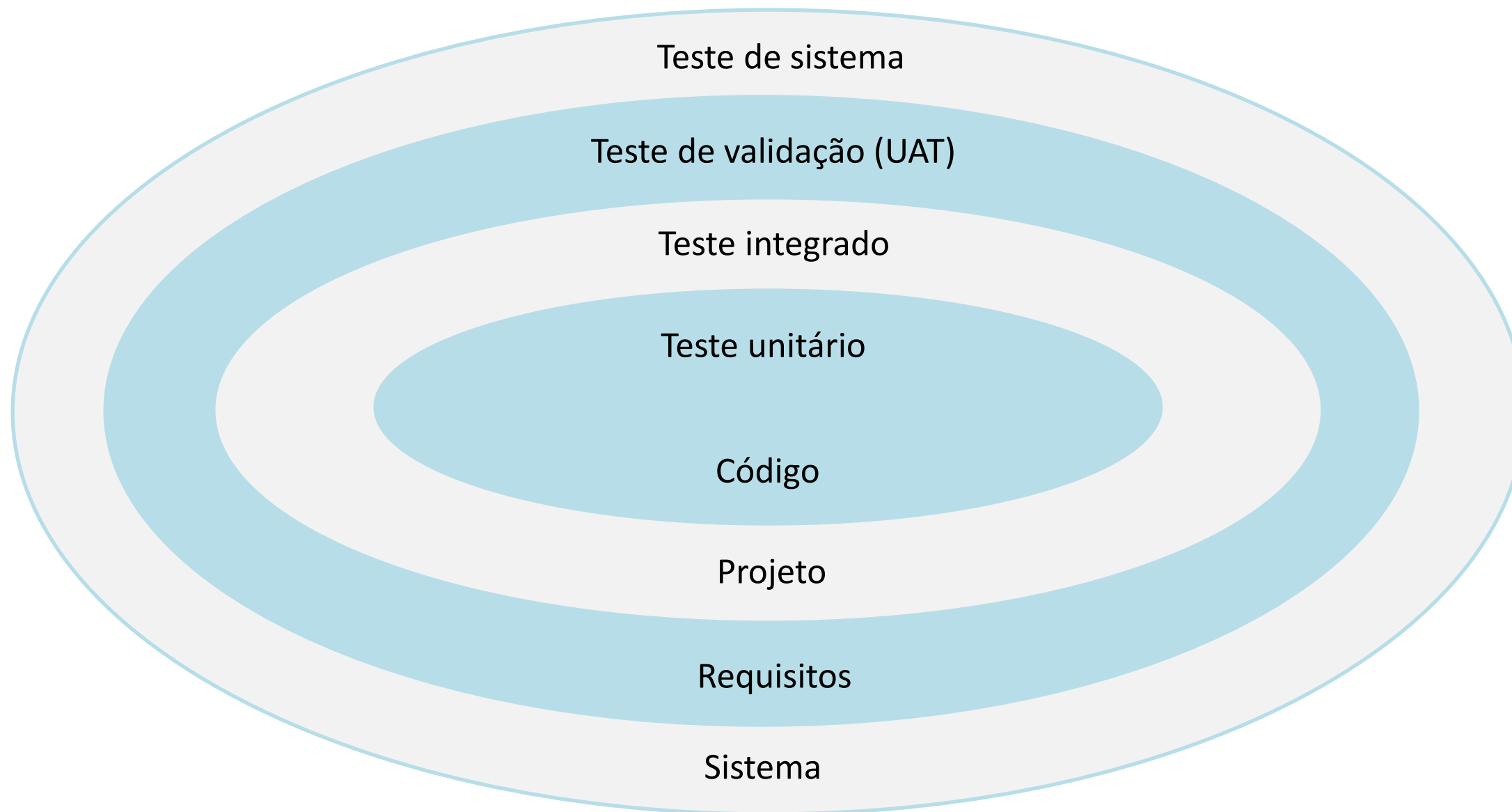
“Teste é o processo de exercitar um programa com a intenção específica de encontrar erros antes de entregar o mesmo para o usuário final.”

Sommerville



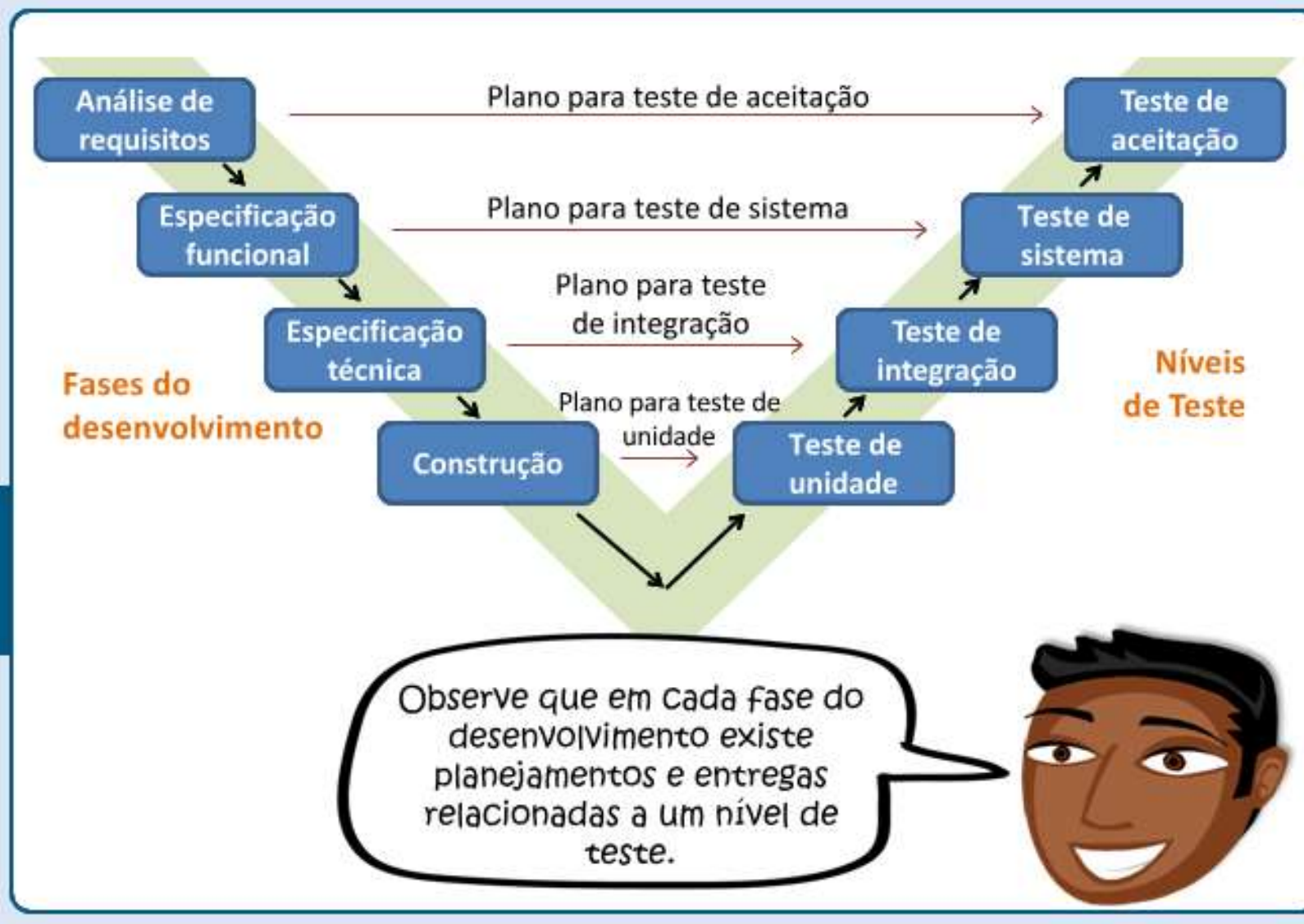
<https://vidaprogramador.com.br/2012/01/10/tester/>

# Níveis de Teste/Estratégia de Teste



# Níveis de Teste/Estratégia de Teste

## Teste *no ciclo de vida*





# Quem testa o software?

Se você acha que é o usuário...meu Deus!

ALFA – Ambiente controlado – dentro de casa – pessoas selecionadas

BETA – Externo com pessoas selecionadas. (innovators)

Normalmente quem são os responsáveis pelos Testes de Software:

- Desenvolvedor/Programador
- Tester
- Muito comum o Analista de Negócios também testar representando o cliente (ele simula também uma pré-validação/pré-homologação)

- Garantir que os problemas serão descobertos cedo!
- Facilitar a manutenção do código
- Servir como documentação
- Ajuda a melhorar o design do código e consequentemente você vira um desenvolvedor melhor
- Reduzir o medo
- Nas linguagens orientadas a objeto, você testa a classe, o método e até o objeto.
- Test Case é o teste para cada classe feito pelo desenvolvedor



# Teste Unitário – O que testar?

- O segredo dos Testes Unitários é saber o que testar, você precisa usar a criatividade para imaginar os testes possíveis, mas com o tempo treinando você consegue evoluir e até gerar uma lista mais comum de possibilidades, afinal, testar divisão por 0 não ocorre uma única vez. 😊
- Não faz sentido testar itens triviais como GET/SET a não ser que você tenha muito código neles;
- Também não faz sentido fazer muitos testes na mesma condição e deixar outras condições para trás;
- Use apenas os dados necessários;
- Comece pelo mais simples e depois vá para o complexo;
- Achou um bug? Escreva classe que testa esse bug e depois conserte;

# Tipos de Testes Integrados

- Integridade da interface: Validar a interface do usuário a cada novo módulo adicionado.
- Validade funcional: Teste para descobrir defeitos funcionais no software
- Conteúdo da informação: Teste de erros em estruturas de dados, ortografia, etc
- Desempenho: Verificar se os limites de desempenho especificados foram testados

## TESTE DE REGRESSÃO:

A medida que novos módulos são inseridos, novos “caminhos” são criados e desta forma é necessário testar novamente um conjunto de cenários para garantir que tudo continua funcionando corretamente.

## TESTE DE FUMAÇA (COMPILAÇÕES DIÁRIAS)

Uma compilação inclui todos os arquivos de dados, bibliotecas, módulos reutilizáveis e componentes projetados necessários para implementar uma ou mais funções do produto. Uma série de testes é projetada para expor erros que impedirão a construção de executar adequadamente sua função.

A intenção deve ser descobrir erros bloqueadores que tenham a maior probabilidade de atrasar as entregas. A construção é integrada a outras construções e todo o produto é testado diariamente, normalmente de forma automatizada.

# Testes de Aplicações Web (Mobile é similar)

1. Conteúdo
2. Banco de Dados
3. Interface do usuário
4. Usabilidade
5. Compatibilidade
6. Componentes
7. Navegação
8. Configuração
9. Segurança
10. Performance
11. Carga
12. Stress



<https://paulomonteiro.wordpress.com/cansado/>

Conteúdo: Avaliado em um nível sintático e semântico.

- nível sintático: a ortografia, a pontuação e a gramática são avaliadas para documentos baseados em texto.
- nível semântico - correção (da informação apresentada), consistência (em todo o objeto de conteúdo e objetos relacionados a informação é a mesma) e falta de ambiguidade.

Um CheckList.

- A informação é factualmente precisa?
- A informação é concisa e direta?
- O layout do objeto de conteúdo é fácil para o usuário entender?
- As informações incorporadas em um objeto de conteúdo podem ser encontradas facilmente?
- Já foram fornecidas referências adequadas para todas as informações derivadas de outras fontes?
- As informações apresentadas são consistentes internamente e consistentes com as informações apresentadas em outros objetos de conteúdo?
- O conteúdo é ofensivo, enganoso ou abre as portas para litígios?
- O conteúdo infringe direitos autorais ou marcas registradas existentes?
- O conteúdo contém links internos que complementam o conteúdo existente? Os links estão corretos?
- O estilo estético do conteúdo está em conflito com o estilo estético da interface?

- Links: Testar se os links estão corretos, ou seja, a navegação.
- Formulários: Testar as Validações de Tela, ou seja, preenchimentos obrigatórios, campos especiais, caracteres especiais, campos incompletos, etc.
- Script: Testar se o JavaScript está funcionando corretamente nos diversos browsers (engines). **(Atenção para também testar itens como "applets")**
- HTML dinâmico: Testar se objetos de conteúdo que são manipulados no lado do cliente usando scripts ou folhas de estilo em cascata (CSS) estão corretos.
- Janelas pop-up do lado do cliente: Testar o comportamento, se estão funcionando e as limitações.
- Streaming de conteúdo: Testes o funcionamento dos vídeos, áudios ou similares.
- Cookies: Testar o funcionamento e validar o conteúdo dos cookies.

## CheckList

- Testar a conectividade com o(s) banco(s) de dados (Driver, versão, etc)
- Validar as Querys
- **Validar o "COLLATION", ou seja, conjunto de caracteres utilizados pelo banco de dados (língua)**
- Testar as cargas e exportações se existirem, assim como as transformações de dados



- Navegadores disponíveis
- Sistemas Operacionais disponíveis: Android, Linux, Windows, IOS, etc.
- Velocidade de conexão com a Internet
- Responsividade
- Plug-ins (Flash, RealPlayer, etc)

# Aplicações Web: Teste de Navegação

- *Links de navegação:* Conforme teste da interface
- *Redirecionamentos:* Como os links entram em ação quando um usuário solicita uma URL inexistente ou seleciona um link cujo destino foi removido ou cujo nome foi alterado. Testar as diversas formas de solicitar a URL (www, direto, etc).
- *Marcadores:* Embora os marcadores sejam uma função do navegador, o WebApp deve ser testado para garantir que um título de página significativo possa ser extraído à medida que o marcador é criado.
- *Quadros e conjuntos de quadros:* Testar o conteúdo correto, layout e tamanho adequados, desempenho de download e compatibilidade com o navegador
- *Mapas do site:* Cada entrada do mapa do site deve ser testada para garantir que o link leve o usuário ao conteúdo ou à funcionalidade adequada.
- *Mecanismos de pesquisa internos:* O teste do mecanismo de pesquisa valida a exatidão e integridade da pesquisa, as propriedades de manipulação de erros do mecanismo de pesquisa e os recursos avançados de pesquisa, ou a integração com a pesquisa externa (Google).

# Aplicações Web: Teste de Configuração do Servidor

- Os servidores estão com todas as bibliotecas, sdks, etc..na VERSÃO CORRETA?
- O WebApp é totalmente compatível com o sistema operacional do servidor ou da cloud?
- As medidas de segurança do sistema (por exemplo, firewalls ou criptografia) permitem que o WebApp execute e atenda os usuários sem interferência ou degradação do desempenho?
- O WebApp foi testado com a configuração do servidor distribuído (se houver) que foi escolhido?
- Os scripts WebApp do lado do servidor são executados corretamente?
- Os logs de erros e da aplicação estão sendo gerados corretamente?
- Se os servidores proxy forem usados, as diferenças em suas configurações foram tratadas com o teste no local?

# Aplicações Web: Outros Testes

- Testes de Segurança: Simulações de ataque, análise de bugs e vulnerabilidades pré-existentes, seja do lado do cliente, seja do lado do servidor. Ex: Firewall, Autenticação, Criptografia.
- Testes de Performance: Medir os tempos de respostas de acesso aos diversos módulos do sistema e entender como isso afeta a utilização do sistema.
- Testes de Carga: Simular a utilização massiva do sistema, quantidade conexões, volume de dados, volume de transações.
- Testes de Estresse: Simular a utilização massiva do sistema mas buscando os limites para entender como se dá a degradação do sistema e o que ocorre quando os limites são atingidos, ou seja, se ocorrem falhas, lentidões, etc.

## Kahoot – Qualidade – Aula 2



# Engenharia de Software

Aula Extra

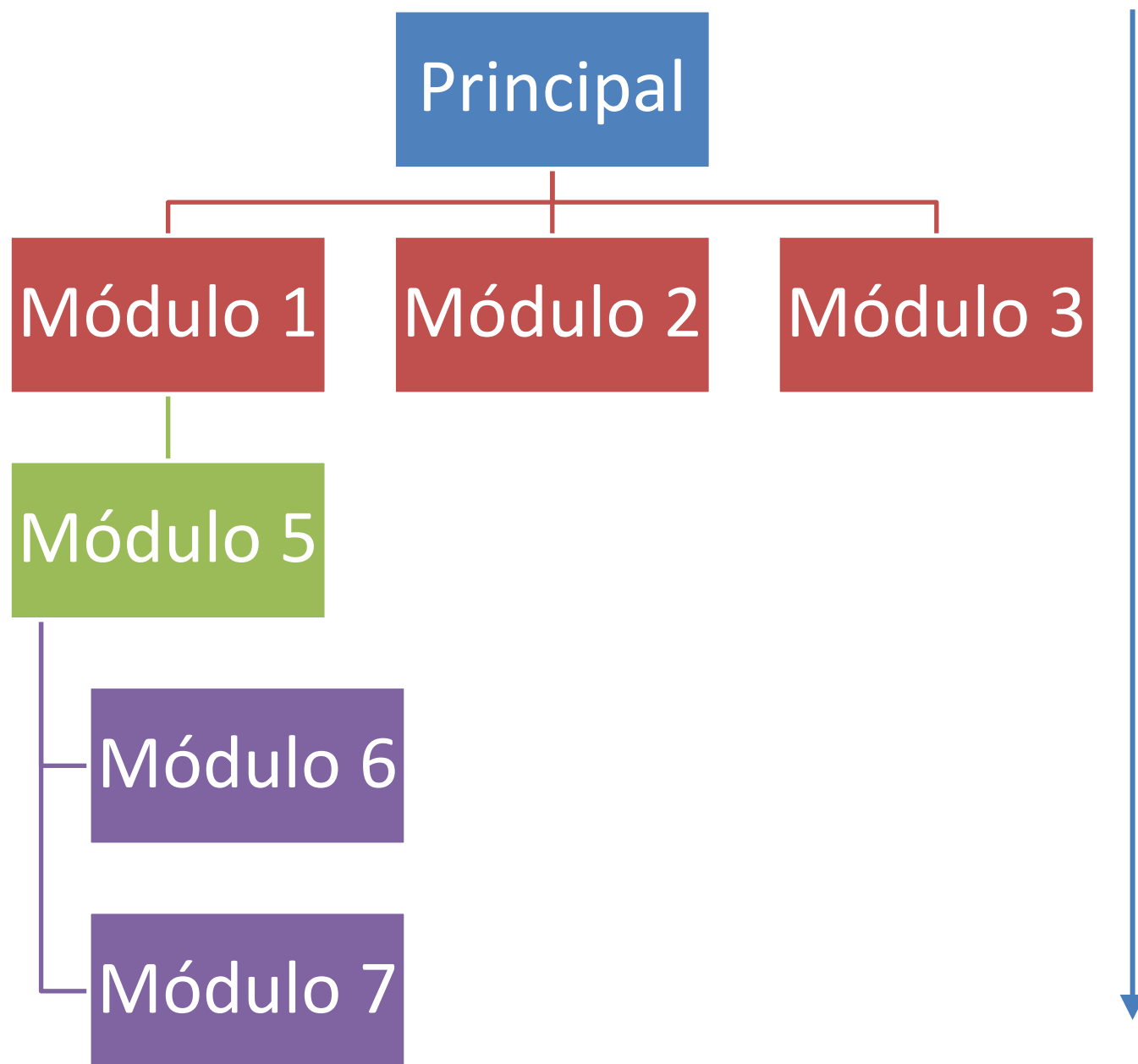
Professor Esp. Gerson Santos

Entender se as partes somadas funcionam corretamente. Caso não seja feita o teste unitário, a complexidade/volume de erros nesta fase pode aumentar muito:

Formas:

- Big Bang
- Incremental

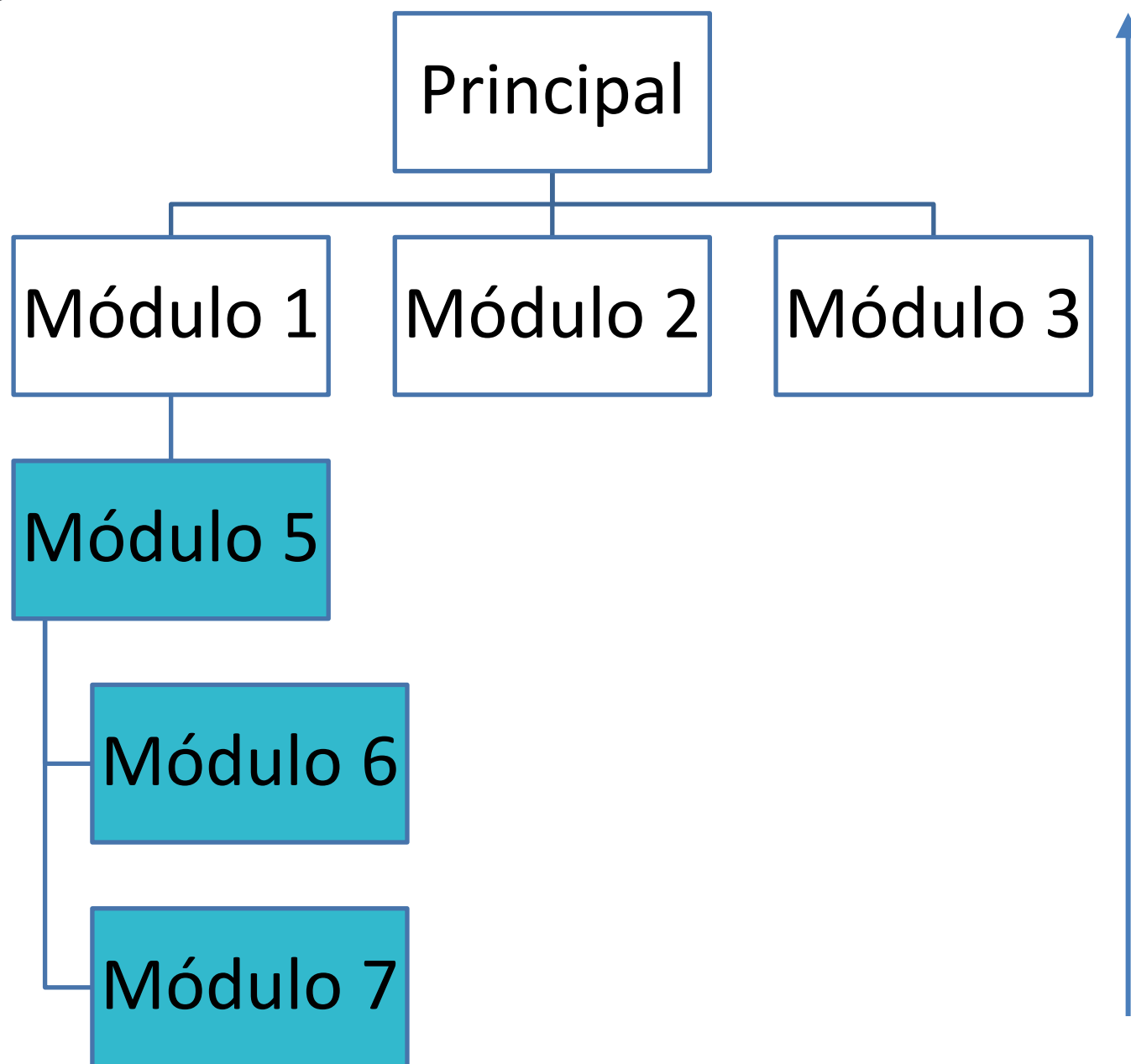
# Teste Integrado – Integração descendente



1. Módulo de Controle Principal é utilizado como um testador
2. Os módulos podem ser substituídos por módulos pseudocontroladores dependendo da abordagem
3. Os testes são realizados a medida que cada módulo é integrado.
4. Quando um novo módulo é integrado um conjunto de testes é realizado novamente (pode ser regressão)

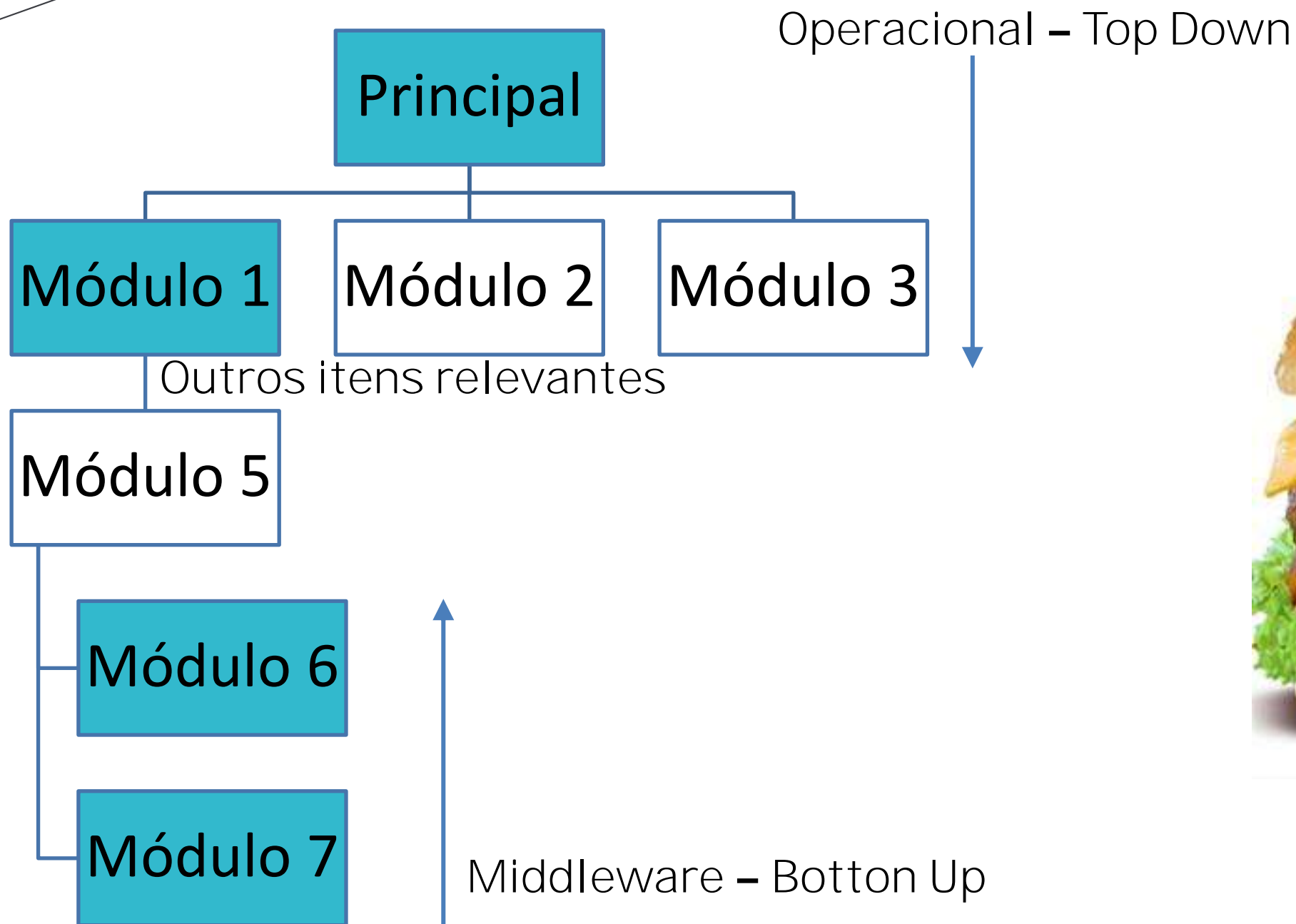


# Teste Integrado – Integração ascendente



1. Os Módulos de baixo nível são agrupados (builds) conforme a função que executam, ou seja, dependências. (Exemplo em azul)
2. Precisa ser escrito um PseudoControlador para coordenar a entrada e saída de teste.

# Teste Integrado – Sanduíche



# Comparação dos Testes

	Bottom-up	Top-down	Big Bang	Sandwich
Integração	Cedo	Cedo	Tarde	Cedo
Tempo para o funcionamento básico do programa	Tarde	Cedo	Tarde	Cedo
Drivers (mocks) – dublê do front	Sim	Não	Sim	Sim (Alguns)
Stubs (mocks) – dublê do back	Não	Sim	Sim	Sim (Alguns)
Trabalho em paralelo no início	Médio	Baixo	Alto	Médio
Capacidade de Testar caminhos particulares	Fácil	Difícil	Fácil	Médio

<https://qualidadebr.wordpress.com/2009/05/10/tecnicas-de-integracao-de-sistema-big-bang-e-sandwich/>

<https://pt.stackoverflow.com/questions/157330/qual-o-conceito-de-stubs-e-de-drivers-em-testes-de-integra%C3%A7%C3%A3o>

# OBRIGADO!