Aula 4 – Estágios ou níveis de teste

Prof. Paulo R. Nietto

Universidade Anhembi Morumbi
LAUREATE INTERNATIONAL UNIVERSITIES

Objetivos

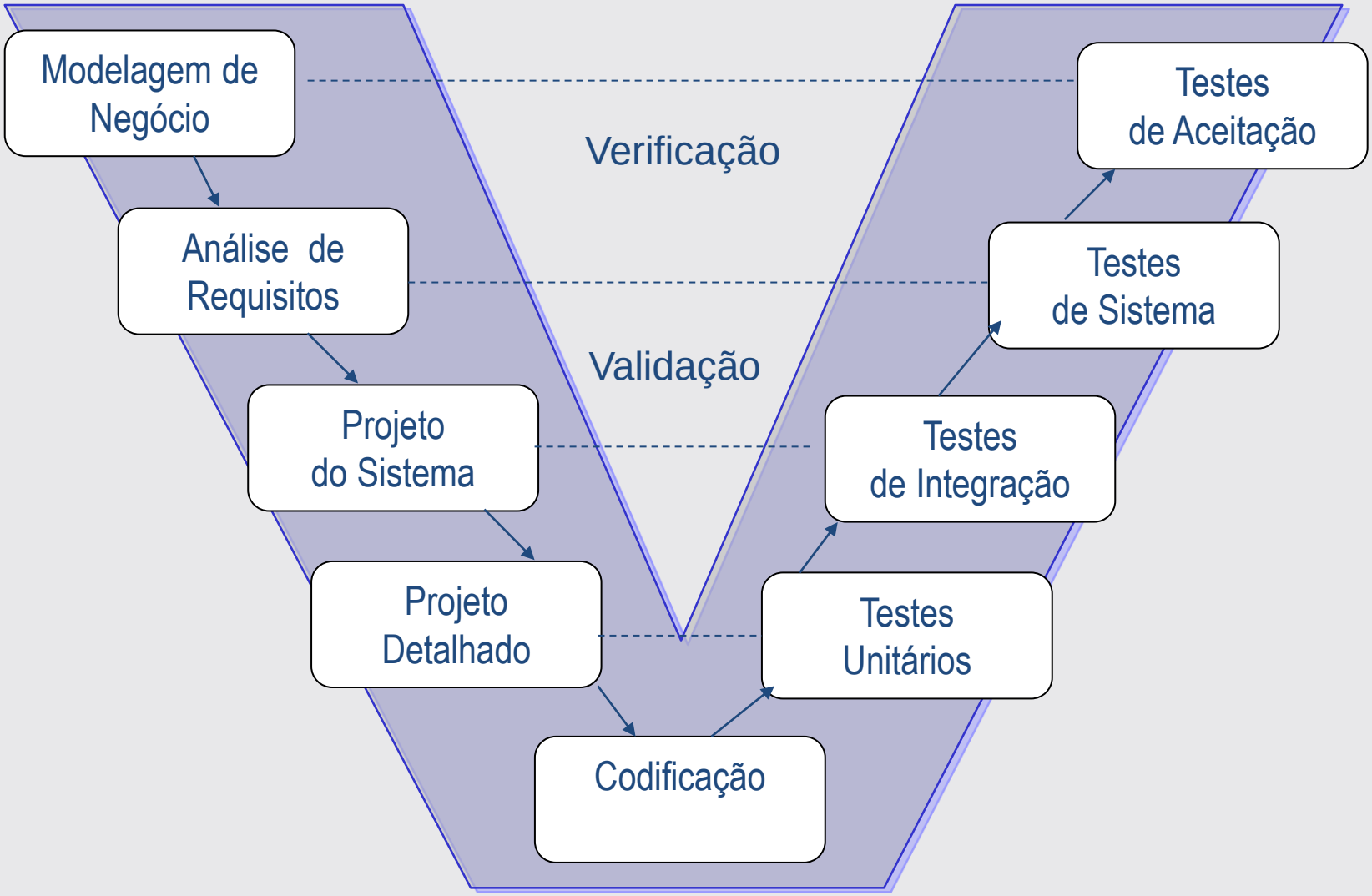
- Apresentar níveis/estágios de teste
- Exercícios

Níveis de teste

- A cada fase do desenvolvimento de software deve-se aplicar um tipo de teste

- Tipos de teste:
  - Teste unitário
  - Teste de integração
  - Teste de sistema
  - Teste de aceitação

Modelo em “V”


Modelagem de Negócio
Análise de Requisitos
Projeto do Sistema
Projeto Detalhado
Codificação
Testes Unitários
Testes de Integração
Testes de Sistema
Testes de Aceitação
Verificação
Validação

Teste unitário

- Também conhecido como teste de componente (muitas vezes considerados como dois tipos de teste diferentes)
- Primeiro nível de teste
- Assegurar que cada funcionalidade especificada para o componente tenha sido implementada corretamente

Teste unitário

- O menor item testável individualmente pode ser: função, classe ou componente

- Enfoque:
  - na lógica interna - funcionalidades
  - estruturas de dados

Objetivos Teste Unitário

- Encontrar falhas (bugs), adquirir confiança e reduzir riscos em partes **individuais** antes que estas partes sejam integradas para compor o sistema.

Teste Unitário - Quem? Quando? Onde?

- Quando realizar o teste unitário?
  - Após o término do desenvolvimento da unidade de programa
- Quem realiza?
  - Desenvolvedor
- Onde é realizado o teste unitário?
  - No ambiente de desenvolvimento

Tipos de problemas que podem ser identificados

- Inicialização incorreta de variáveis
- Operadores ou precedência artimética incorretas
- Falta de precisão
- Comparações com tipos de dados diferentes
- Tipos de dados incorretos
- Comparação de variáveis incorretas
- Loops incorretos ou infinitos
- Alterações indevidas no conteúdo de variáveis
- Uso de recursos de memória
- Acessos a rede
- Etc...

Teste Unitário

- Quando detectada uma falha, a correção é feita sem a necessidade de qualquer relato oficial

Teste Unitário

- Ideal é que um componente seja desenvolvido para fazer uma única função específica (alta coesão)

- Facilita a elaboração dos testes e descoberta de erros mais rápida

Teste Unitário

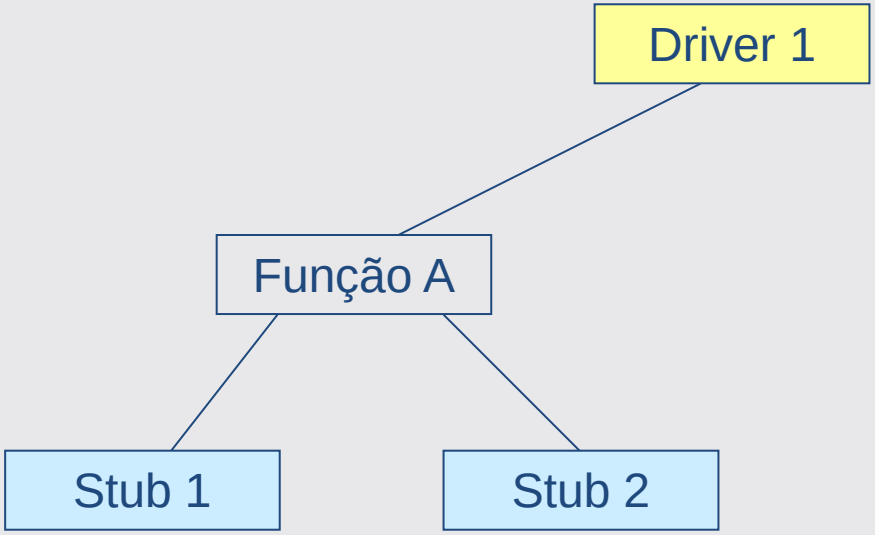
- Alguns itens para serem testados dependem da utilização/integração com outros componentes

- O que fazer quando estes componentes não estão finalizados?

Aplicando teste unitário

- **Driver** – controlador – função que envia dados ao módulo a ser testado (exemplo: main)
- **Stub** – controlado – função chamada pelo módulo que esta sendo testado
- Simulam o fluxo de informações entre o módulo em teste e os demais a serem integrados

Aplicando teste unitário



- Gera aumento do custo do projeto
- Devem ser o mais simples possível
- Devem ser desenvolvidos para o teste em questão
- O teste adequado do componente só é possível durante os testes de integração

Teste de Integração

‘Se todos os componentes funcionam individualmente, por quê não vão funcionar quando estiverem todos juntos??? ‘

Teste de Integração

- Segundo nível de testes
- Componentes construídos e testados separadamente
- Ao serem agrupados deve-se verificar se eles interagem de forma correta
- **Enfoque é verificar se componentes se comunicam conforme especificado**

Objetivos Teste de Integração

- Encontrar falhas (bugs), adquirir confiança e reduzir riscos na **comunicação** entre os componentes quando unidos para formar o sistema em questão

- **Auxilia na construção da arquitetura do software buscando erros associados às interfaces**

Teste de Integração

- A maior dificuldade em colocar todos os módulos juntos esta na **interface** entre eles:
  - Dados podem ser perdidos
  - Um componente pode ter efeito imprevisto ou adverso sobre outro
  - Componentes em conjunto podem não produzir a funcionalidade desejada...

Teste de Integração - Quem? Quando? Onde?

- Quando realizar o teste integração?
  - Em componentes que devem ser integrados e que já tenham passados pelo teste unitário
- Quem realiza?
  - Desenvolvedores, Testadores, Analistas de Sistemas, DBA, etc.
- Onde é realizado o teste unitário?
  - No ambiente de desenvolvimento

Níveis de teste de integração

- Pode haver mais de um nível de teste de integração:
  - Teste de Integração entre componentes
  - Teste de Integração entre sistemas – integração entre sistemas. Realizado **após o teste de sistema**

Métodos de integração

- Big-Bang
- Incremental:
  - Top-Down
  - Bottom-Up

Métodos de integração

- **Big-Bang:** todos os componentes já testados são combinados de uma única vez e testados em conjunto
  - Caos!
  - Muitos erros são encontrados: qual a causa? onde ocorreu?
  - Correção pode gerar outros erros...

Métodos de integração

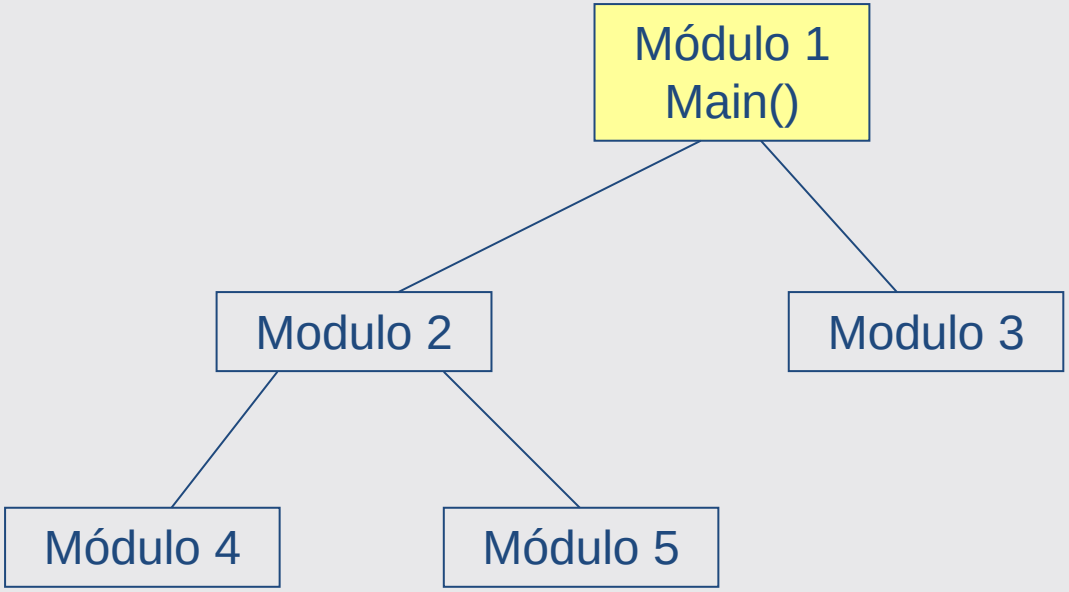
- **Incremental:** os componentes são desenvolvidos e testados em partes
  - Maior facilidade para encontrar erros
  - Maior probabilidade de ser testado por completo

Incremental – Top Down

- A integração dos componentes ocorre de cima para baixo

- Iniciando-se com o programa principal (main) e acrescentando-se os módulos subordinados

Incremental – Top Down

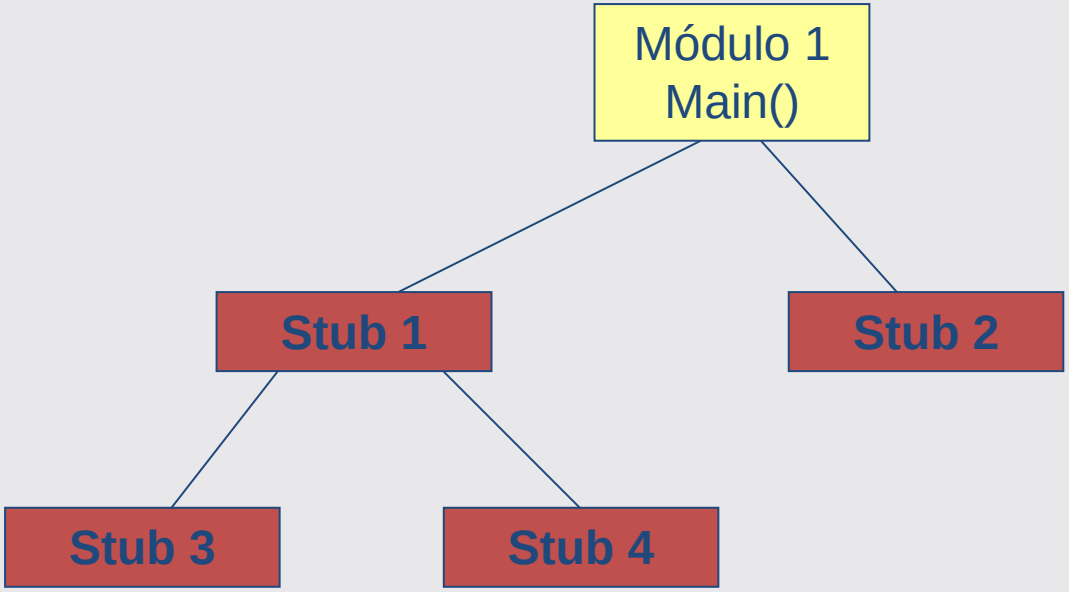


Incremental – Top Down

- Inicia através do teste do programa principal e utiliza **stubs*** para os níveis abaixo
- Cada stub vai sendo substituído por um componente
- Módulos são integrados um a um e testados
- Processo se repete até toda estrutura do programa esteja construída

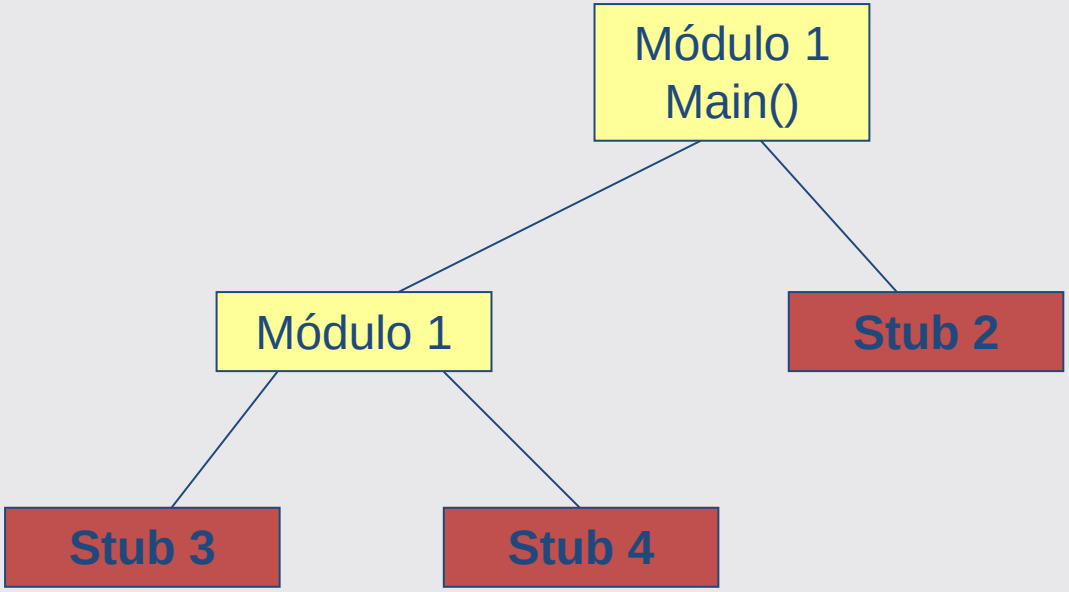
Stubs= pequenos códigos utilizados somente temporariamente para simular uma funcionalidade ainda inexistente

Incremental – Top Down

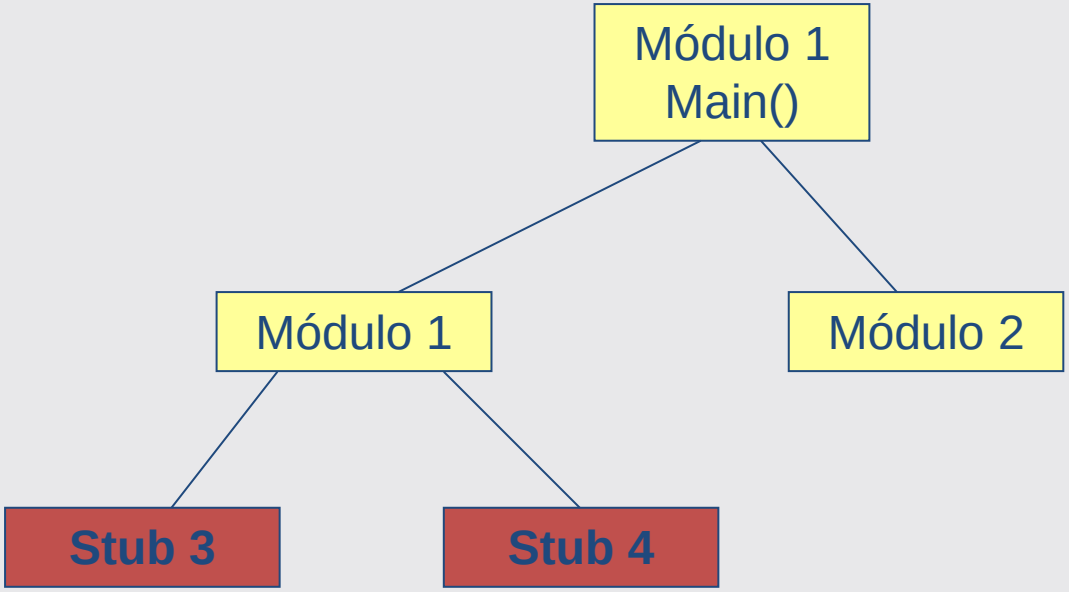


Inicia pelo Modulo Principial e usar stubs para testar os modulos inferiores

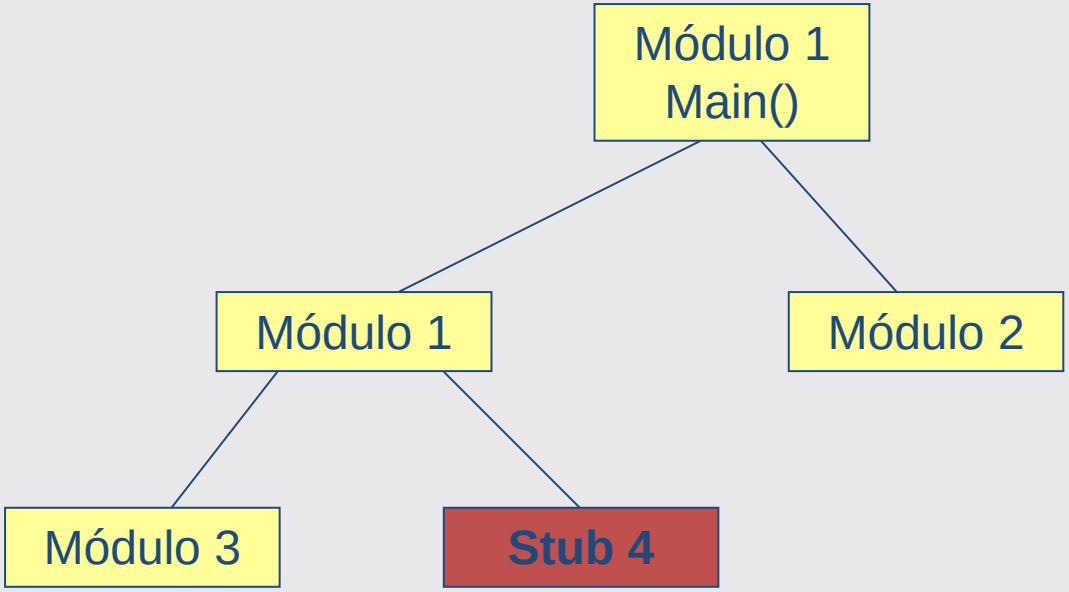
Incremental – Top Down



Incremental – Top Down



Incremental – Top Down



Pontos a considerar na abordagem Top Down

- Problemas de arquitetura podem ser identificados mais cedo
- Mais apropriado para desenvolvimento estruturado
- Necessidade de grande infra-instrutura
- Grande utilização de stubs – Alto custo

Incremental – Top Down

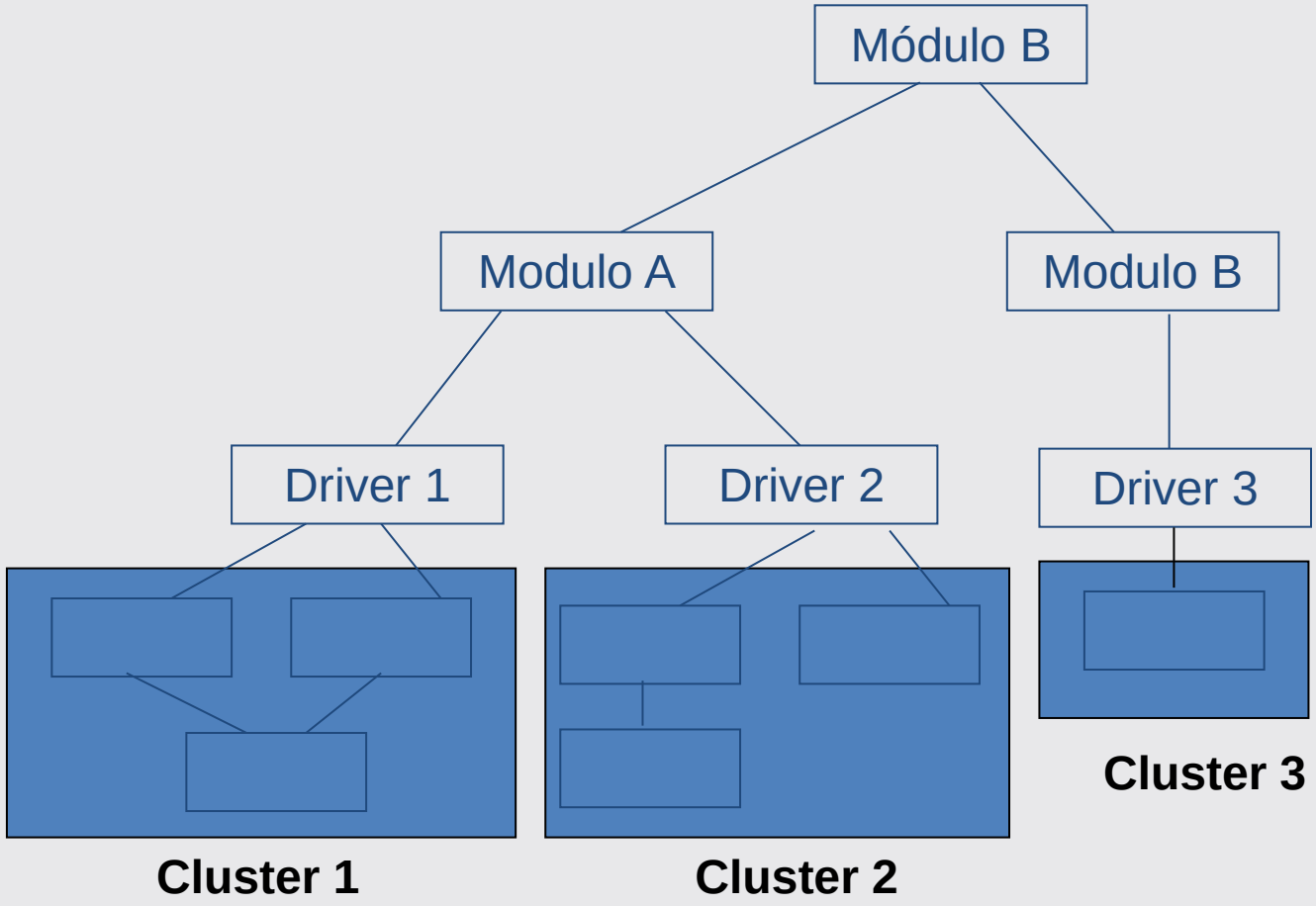
- Problemas:
  - O que acontece se os problemas estiverem nos stubs da camada mais baixa?
    - Demora na detecção de erros nos baixos níveis

Incremental – Bottom-up

- A integração dos componentes ocorre de baixo para cima

- Iniciando-se com a construção e teste de módulos da estrutura do programa nos níveis mais baixos

Incremental – Bottom-up

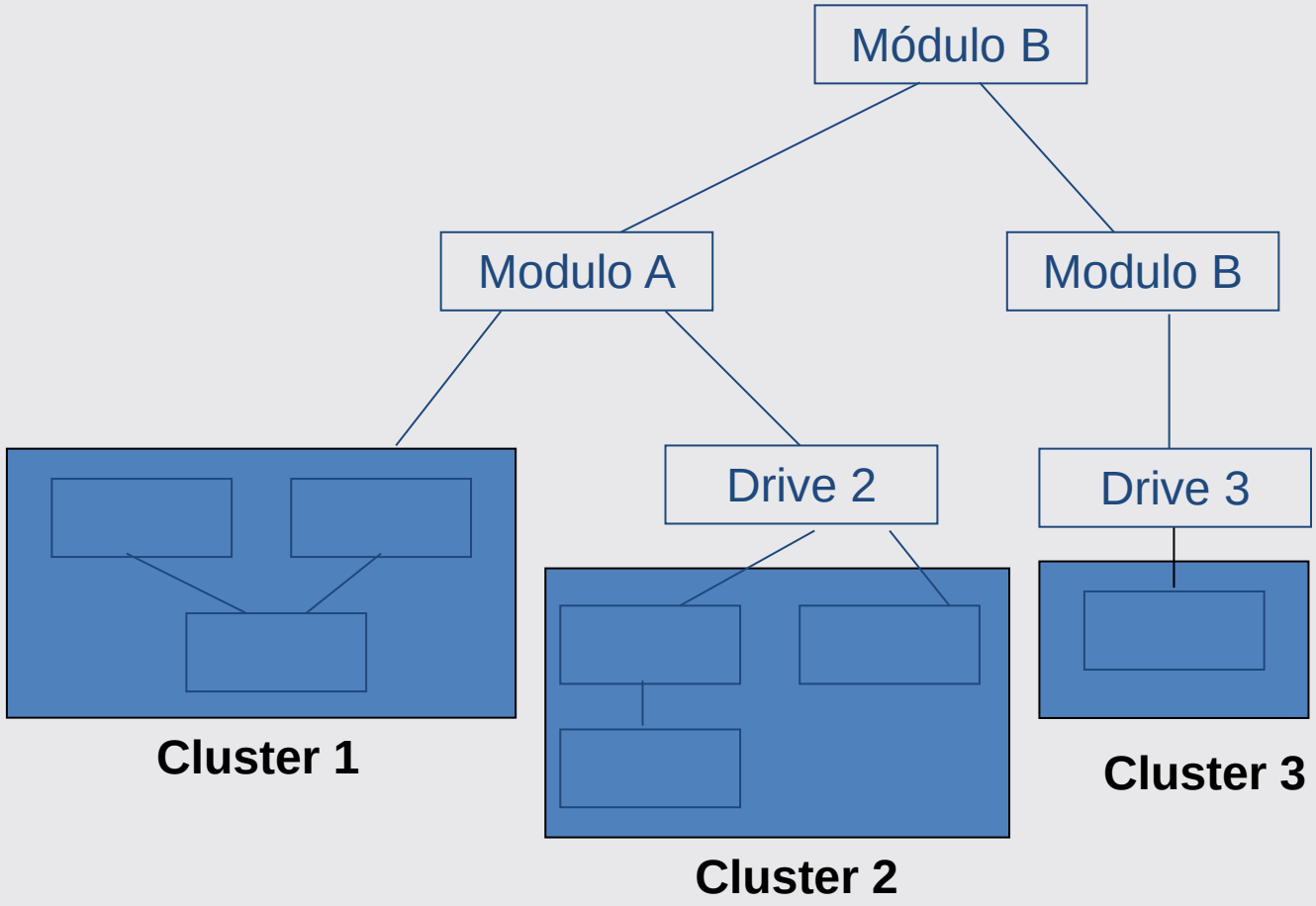


Incremental – Bottom-up

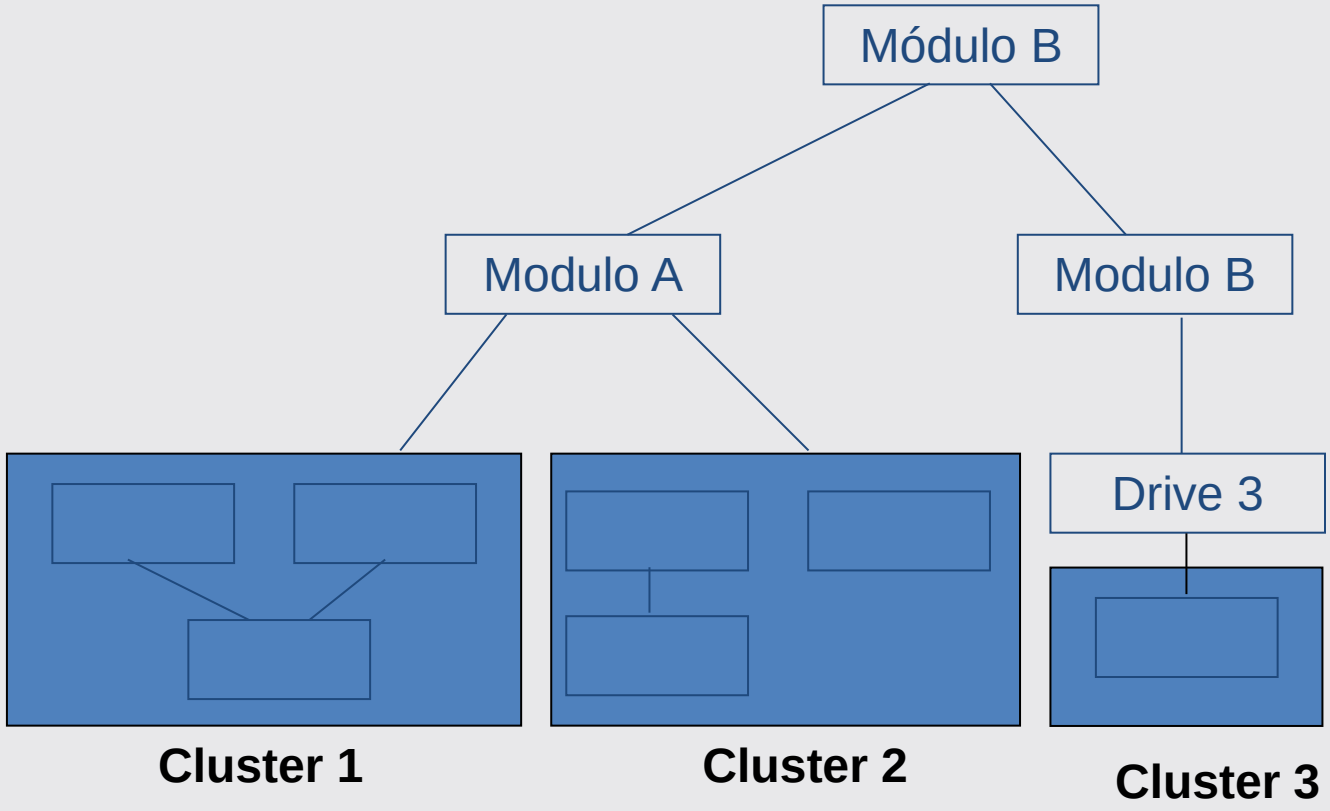
- Utiliza **Drivers*** para os níveis acima
- Componentes de baixo nível são **agregados** em **cluters** de acordo com subfunções específicas que realizem
- Um Driver é acrescentado para coordenar a entrada e saída dos dados do **cluster**
- O **cluster** gerado é testado e o driver é removido e substituído pelo módulo(s) acima
- Processo se repete até toda estrutura do programa esteja construída

***Driver = código que invoca as rotinas de baixo nível, testando-as com diversas combinações de parâmetros**

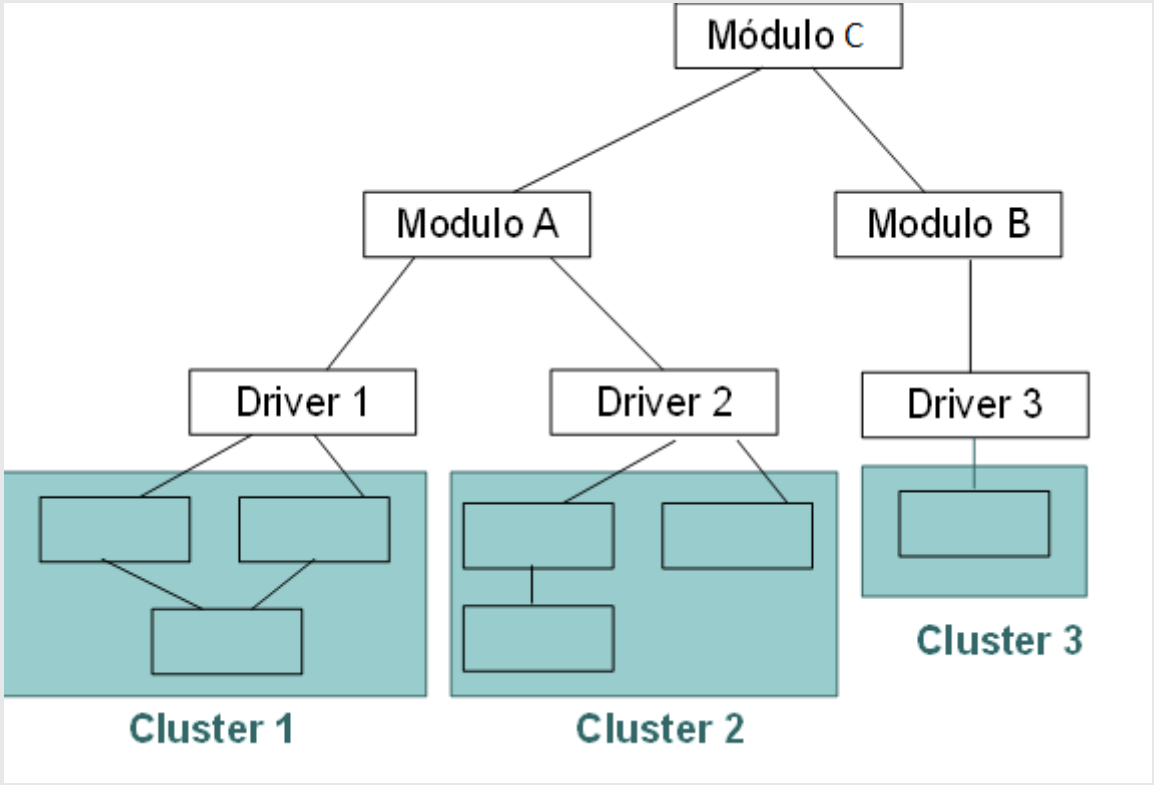
Incremental – Bottom-up



Incremental – Bottom-up



Incremental – Bottom-up



Incremental – Bottom-up

- Problemas:
  - O que acontece se os problemas estiverem nos stubs da camada mais alta?
    - Problemas nos níveis altos detectados tardiamente

Pontos a considerar na abordagem Bottom-up

- Apropriado para o desenvolvimento orientado a objetos

- Consegue identificar problemas de arquitetura muito tarde

Teste de Regressão

- Cada vez que um novo módulo é adicionado durante o teste de integração o software se modifica

- Como garantir que o que foi testado anteriormente continua correto?

Teste de Regressão

- Tem como meta reduzir “efeitos colaterais”
- Garantir que modificações não introduzirão erros adicionais no software
- Re-executando alguns casos de testes (ou utilizar alguns testes automatizados) que já foram feitos com sucesso antes e comparar os resultados

Teste de Sistema

- Terceiro nível de teste

- Foco no **sistema como um todo**, para validar a **exatidão e perfeição** na execução das **funções requeridas**

Teste de Sistema

- Acontece após todos os testes de integração
- Arquitetura do sistema esta completa ou muito perto de estar completa
- Conhecido como “teste dedo-duro”

Objetivos do Teste de Sistema

- Encontrar falhas (bugs), adquirir confiança e reduzir riscos no **comportamento global** e **particular do sistema**.

Teste de Sistema

- Ambiente de teste deve corresponder o máximo possível ao ambiente de produção
- Ambiente deve conter hardware e software que serão usados com o sistema
- Um erro comum: fazer o teste de sistema no ambiente do cliente!

Teste de Sistema

- Realizados com base em:
  - Especificações de requisitos
  - Especificações de riscos
  - Processos de negócio
  - Casos de uso
  - E outras descrições de alto nível do comportamento, interações e recursos do sistema

Teste de Sistema - Quem? Quando? Onde?

- Quando realizar o teste de sistema?
  - Após o término da integração dos componentes
- Quem realiza?
  - Analista de teste
- Onde é realizado o teste de sistema?
  - No ambiente de teste

Deve tratar de requisitos funcionais e não funcionais

- Desempenho
- Volume
- Documentação
- Robustez

Tipos de Teste de Sistema

- Teste de Recuperação
- Teste de Segurança
- Teste de Estresse
- Teste de Desempenho

Teste de Recuperação

- Forçar o sistema a falhar de diversos modos e verificar se a recuperação é realizada de forma adequada:
  - Dentro do período de tempo especificado
  - Sem prejuízos
  - etc

Teste de Recuperação

- Recuperação automática (pelo próprio sistema) – avaliar o sistema quanto a correção:
  - Re-inicialização
  - Mecanismos de verificação
  - Recuperação dos dados

- Recuperação por intervenção humana:
  - Verificar se tempo médio para reparo esta dentro de limites aceitáveis

Teste de Segurança

- Verifica se os mecanismos de segurança vão protegê-lo durante invasão imprópria:
  - Hackers
  - Empregados descontentes
  - Pessoas desonestas a procura de ganhos ilícitos
  - Etc...

Teste de Segurança

- O Tester realiza o(s) papel(eis) de invasor:
  - Vale tudo!
  - Uso de senha de funcionários
  - Uso de softwares para invasão
  - Causar falhas para tentar invadir o sistema durante a recuperação
  - Etc..

Teste de Segurança

- Com tempo e recursos, o Tester vai acabar invadindo o sistema

- **Objetivo do projetista de sistema**: tornar o custo da invasão maior que o valor da informação que será obtida

Teste de Estresse

- Submeter o sistema a situações anormais:
  - Velocidade de entrada dos dados aumentada para verificar como as funções vão reagir
  - Casos de teste que exigem um máximo de memória ou outros recursos
  - Casos de teste que possam provocar problemas de gestão de memória
  - Etc.

- Até onde o sistema vai aguentar?

Teste de Desempenho

- Testar o desempenho do software durante a execução
- Ocorre em todos os níveis de teste
- O verdadeiro desempenho só pode ser medido com o sistema totalmente integrado!

Teste de Desempenho

- Normalmente acoplados aos testes de estresse

- Uso de instrumentos específicos:
  - medir utilização de recursos
  - medir intervalos de execução
  - registrar eventos
  - retirar amostras do estado da máquina

Teste de Aceitação

- Último nível de testes: antes da implantação em produção

- Verificar se que o sistema está pronto para ser implantado em produção, com base nos critérios de aceitação

Teste de Aceitação

- O software está apto a ser utilizado executando funções e tarefas a que se propõem?

Teste de Aceitação - Quem? Quando? Onde?

- Quando realizar o teste de aceitação?
  - Após o término dos testes de sistema
- Quem realiza?
  - Usuário/Responsáveis pelo sistema
- Onde é realizado o teste de aceitação?
  - No ambiente de aceite/homologação ou de produção

Tipos de Teste de Aceitação

- Teste de Aceitação do Usuário
- Teste Operacional
- Teste de Contrato e Regulamento
- Teste Alfa e Teste Beta

Teste de Aceitação do Usuário

- Usuário conhecedor do negócio verifica se o sistema está apropriado para o uso
- Apropriado quando: quem vai usar não é quem esta pagando pelo sistema
- Havendo grupos de usuários diferentes TODOS devem participar do teste
  - Sistema pode ser rejeitado mesmo estando sem falhas e cumprindo funcionalidades

Teste de Aceitação do Usuário

Qual vocês acham que é o maior problema no teste de aceitação do usuário?

Na visão de alguns usuários, esse software pode ter como propósito automatizar sua função, portanto, sabotagens ao sistema podem ser algo comum nessa etapa.

Teste Operacional

- Teste conduzido para avaliar um componente ou sistema em seu ambiente operacional

- O administrador do sistema avaliará:
  - Teste de Backup/Restore – banco de dados
  - Recuperação de desastre
  - Gerenciamento de usuário
  - Tarefas de manutenção
  - Checagens de vulnerabilidade de segurança

Teste de Contrato e Regulamento

- Verificar a conformidade aos requisitos, regulamentos, ou normas contratualmente acordados ou de exigência legal

Teste Alfa e Beta

- Quando o software é desenvolvido para ser um produto a ser usado por vários clientes, é inviável realizar testes de aceite com cada um.

- Opção: teste de campo para obter feedback de clientes em potencial

Teste Alfa

- Ocorre na empresa fabricante do produto
- Executado pelo cliente
- Presença de desenvolvedores que registram os erros/problemas encontrados

Teste Beta - Teste de Campo

- Ocorre nas instalações do cliente em potencial
- Executado pelo cliente
- Desenvolvedor normalmente não esta presente
- Cliente registra todos os problemas

Teste Alfa e Beta

- Lançamento de pré-versões
- Clientes devem representar adequadamente o mercado em potencial
- Clientes fornecem feedback sobre problemas, comentários e impressões sobre o produto

Exercícios

1. Testar interfaces entre componentes é objetivo de qual nível de testes?

a) Teste de componentes

b) Teste de integração

c) Teste de sistema

d) Teste de aceitação

Exercícios

1. Testar interfaces entre componentes é objetivo de qual nível de testes?

a) Teste de componentes

b) Teste de integração

c) Teste de sistema

d) Teste de aceitação

Exercícios

2. Como exemplo métodos de integração, não se enquadra:

a) Baseado na modelagem

b) Big-Bang

c) Bottom-up

d) Top-Down

Exercícios

2. Como exemplo métodos de integração, não se enquadra:

a) Baseado na modelagem

b) Big-Bang

c) Bottom-up

d) Top-Down

Exercícios

3. Qual a diferença entre re-testar e testes de regressão?

Exercícios

3. Qual a diferença entre re-testar e testes de regressão?

R. O teste de regressão tem como objetivo garantir que modificações não introduzirão erros adicionais no software, enquanto re-testar significa simplesmente testar novamente algo.

Exercícios

4. O que é teste operacional?

Exercícios

4. O que é teste operacional?

R. Teste conduzido para avaliar um componente ou sistema em seu ambiente operacional

Exercícios

5. Não faz parte teste de aceite:
   a) Teste de componente
   b) Teste realizado pelo usuário
   c) Teste baseado em requisitos
   d) Teste de segurança

Exercícios

5. Não faz parte teste de aceite:
   a) Teste de componente
   b) Teste realizado pelo usuário
   c) Teste baseado em requisitos
   d) Teste de segurança

Exercícios

6. Teste unitário e depuração de código podem ser considerados a mesma coisa?

Exercícios

6. Teste unitário e depuração de código podem ser considerados a mesma coisa?

R. Não pois o teste unitário tem como objetivo encontrar falhas (bugs), adquirir confiança e reduzir riscos em partes **individuais** antes que estas partes sejam integradas para compor o sistema, enquanto a depuração visa identificar e remover uma falha.

Exercícios

7. Não faz parte do teste de sistema:
    a) Testar funcionalidades
    b) Testar interações entre componentes
    d) Testar desempenho do sistema
    e) Testar segurança do sistema

Exercícios

7. Não faz parte do teste de sistema:
    a) Testar funcionalidades
    b) Testar interações entre componentes
    d) Testar desempenho do sistema
    e) Testar segurança do sistema

Exercícios

8. Beta teste:
    a) Realizado pelo cliente no seu próprio ambiente
    b) Realizado pelos clientes em seu ambiente de desenvolvimento de software
    c) Realizado por uma equipe de testadores independentes
    d) Útil para testar software feito sob encomenda

Exercícios

8. Beta teste:
   a) Realizado pelo cliente no seu próprio ambiente
   b) Realizado pelos clientes em seu ambiente de desenvolvimento de software
   c) Realizado por uma equipe de testadores independentes
   d) Útil para testar software feito sob encomenda

Exercícios

9. Principal foco do teste de aceitação:
   a) Encontrar falhas no sistema
   b) Garantir que o sistema é aceitável para todos os usuários
   c) Testar a integração do sistema com outros sistemas
   d) Testar uma necessidade de negócio

Exercícios

9. Principal foco do teste de aceitação:
   a) Encontrar falhas no sistema
   b) Garantir que o sistema é aceitável para todos os usuários
   c) Testar a integração do sistema com outros sistemas
   d) Testar uma necessidade de negócio

Exercícios

10. Marque a alternativa falsa:

a) Teste de componente não deve ser executado por desenvolvedores

b) Teste de componente pode ser executado por testers

c) Teste de componente pode necessitar de conhecimento especial do desenvolvedor

d) Teste de componente é o nível mais precoce de teste

Exercícios

10. Marque a alternativa falsa:

a) Teste de componente não deve ser executado por desenvolvedores

b) Teste de componente pode ser executado por testers

c) Teste de componente pode necessitar de conhecimento especial do desenvolvedor

d) Teste de componente é o nível mais precoce de teste

Referências

- SOMMERVILLE, Ian. Engenharia de Software. São Paulo: Addisson-Wesley, 2004
- PRESSMAN, Roger S. Engenharia de Software. São Paulo: Makron Books, 1995.
- CRAIG, R.D. Systematic Software Testing. New York: Artech House, 2002.
- Jeff Tian. Software Quality Engineering - Testing, Quality Assurance, and Quantifiable Improvement. IEEE Computer Society. John Wiley & Sons, Inc. 2005.

Obrigado

paulo.nietto@animaeducacao.com.br

Universidade
Anhembi Morumbi
LAUREATE INTERNATIONAL UNIVERSITIES