

Testes de Software

Fases

Baseado em notas de aula da profa. Eliane Martins

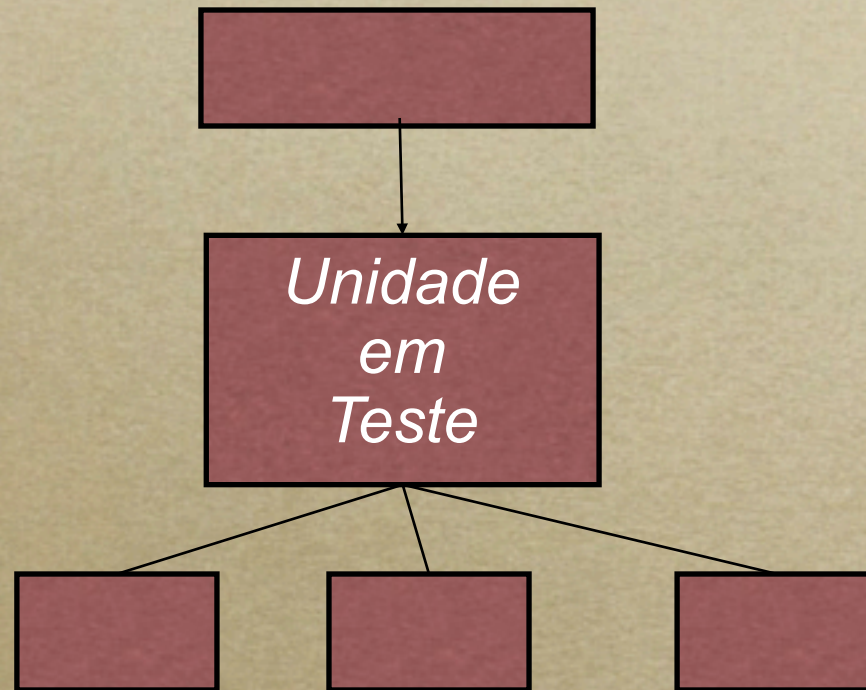
Tópicos

- *Testes de Unidades*
- *Testes de Integração*
- *Testes de Aceitação e de Sistemas*
- *Testes de Regressão*

Testes de Unidades

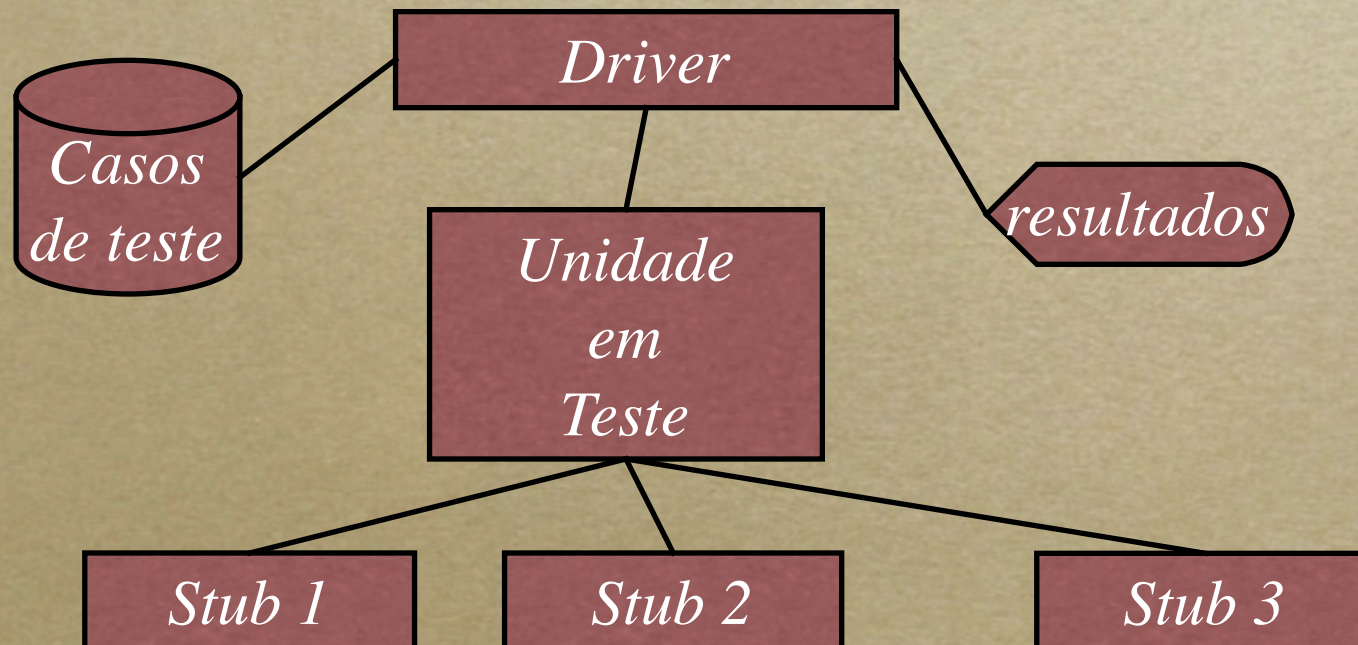
- *Visam exercitar detalhadamente uma unidade do sistema:*
 - *módulo ou função; classe ou pequenos “clusters”*
- *Aspectos a serem considerados :*
 - *interfaces: verificar parâmetros de entrada e saída*
 - *estruturas de dados: verificar integridade dos dados armazenados*
 - *condições de limite: verificar se a unidade opera adequadamente nos limites estabelecidos*
 - *tratamento de erros*

Componentes de testes-1



Componentes de testes-2

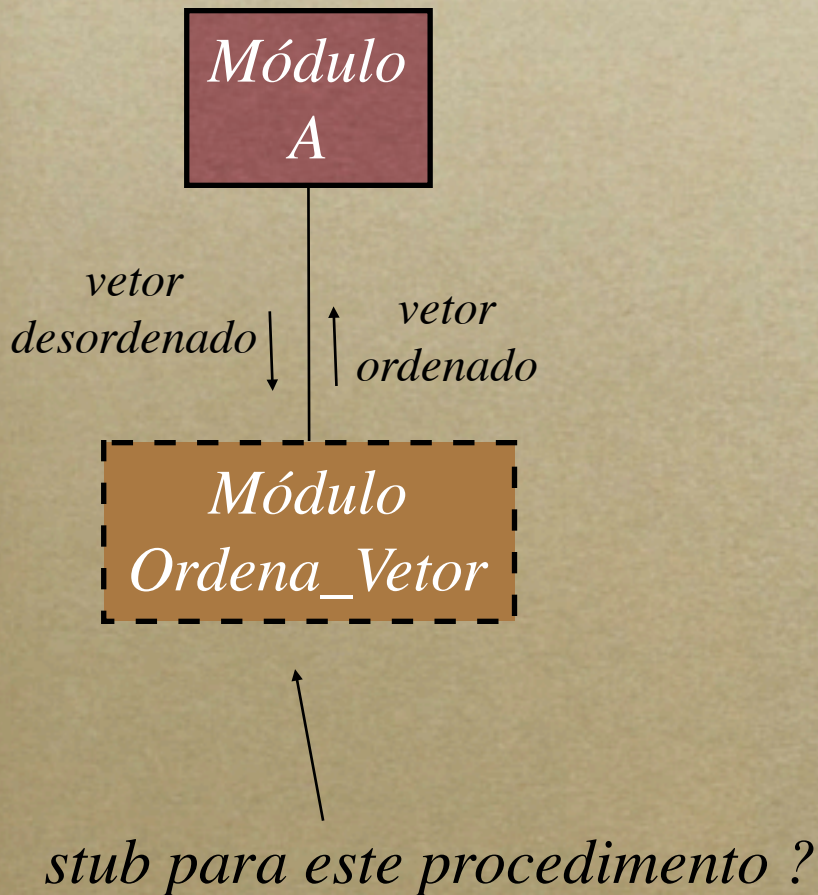
- *Necessários quando se testam unidades isoladamente. Podem ser:*
 - *driver (controlador): chama a unidade em teste*
 - *stub: substitui uma unidade que é chamada*



Componentes de testes

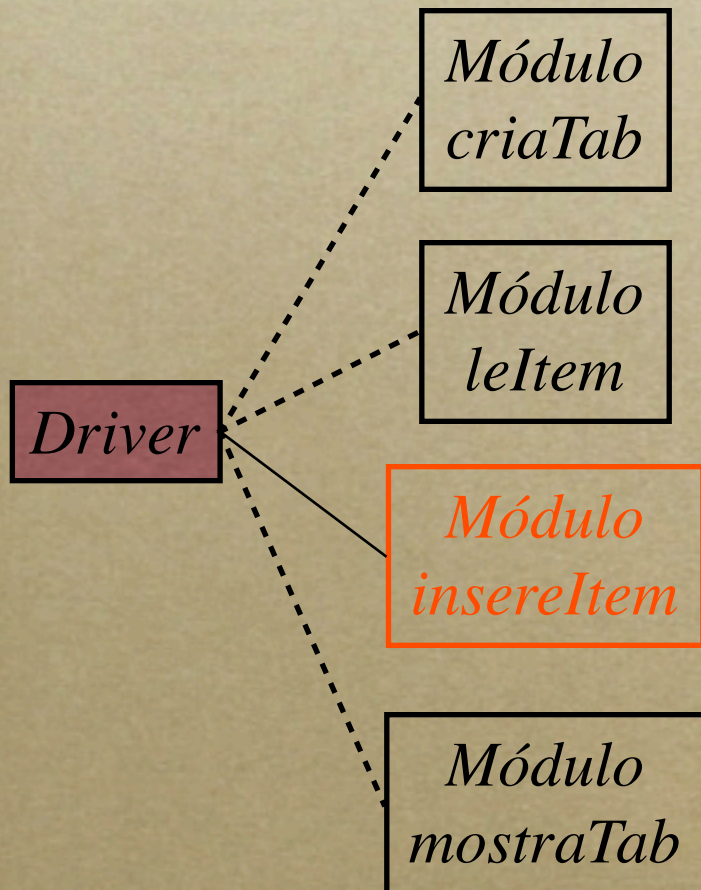
- *devem ser mais simples e mais rápidos de desenvolver do que as unidades substituídas*
- *grau de facilidade ou dificuldade de construí-los depende da qualidade do projeto*

Exemplo: stub



```
type VetorInt = array [1 .. N] of
    integer;
...
procedure Ordena_Vetor (a : VetorInt);
...
begin
    write ("Valores fornecidos");
    for i := 1 to N do write (a [ i ] );
    write ("Forneça os valores
ordenados");
    for i := 1 to N do read (a [ i ] );
end;
```

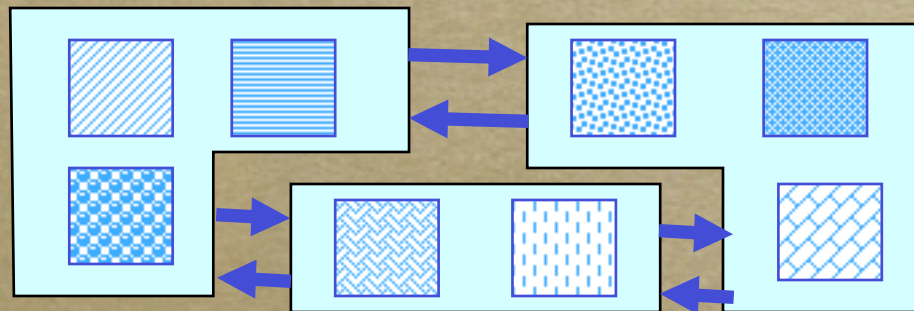
Exemplo - Driver



```
type TabInt = array [ 1 .. N, 1 .. M ] of  
    integer;  
...  
var Tabela: TabInt,  
    x: integer;  
...  
criaTab ;  
leItem ( x );  
insereItem ( x );  
mostraTab ;  
....
```

Testes de integração

- *Integram unidades já testadas*
- *critério: exercitar cada unidade pelo menos uma vez*



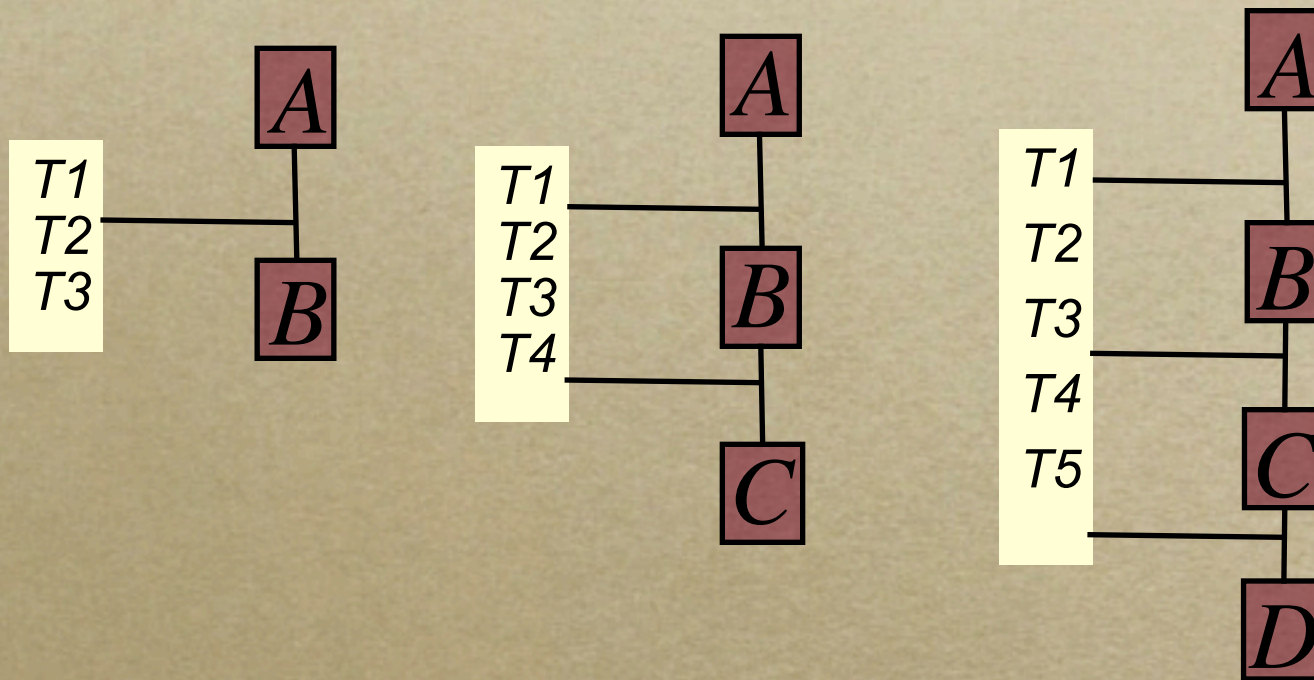
Falhas de integração

- *interfaces incorretas*
- *falta, sobreposição ou conflito de funcionalidades*
- *violação da integridade de arquivos e estruturas de dados globais*
- *seqüência incorreta de unidades*
- *tratamento de erros (exceções) incorreto*
- *problema de configuração / versões*
- *falta de recursos para atender a demanda das unidades*
- *objeto incorreto é associado a mensagem (polimorfismo)*

Abordagens de integração

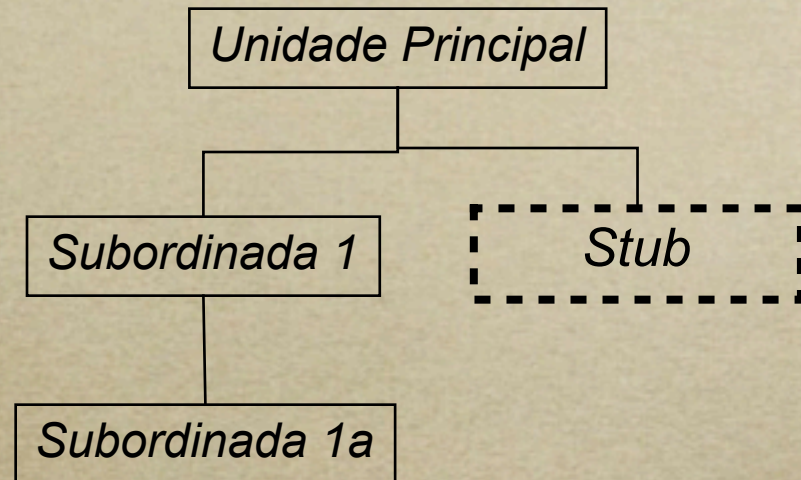
- *Não incremental (“big-bang”):*
 - *todas as unidades são integradas de uma só vez*
 - *esforço de preparação menor*
 - *esforço para diagnóstico e correção de falhas é maior*
- *Incremental*
 - *as unidades são integradas gradualmente*
 - *descendente (“top-down”)*
 - *ascendente (“bottom-up”)*
 - *por colaboração*
 - *mista*

Abordagem incremental



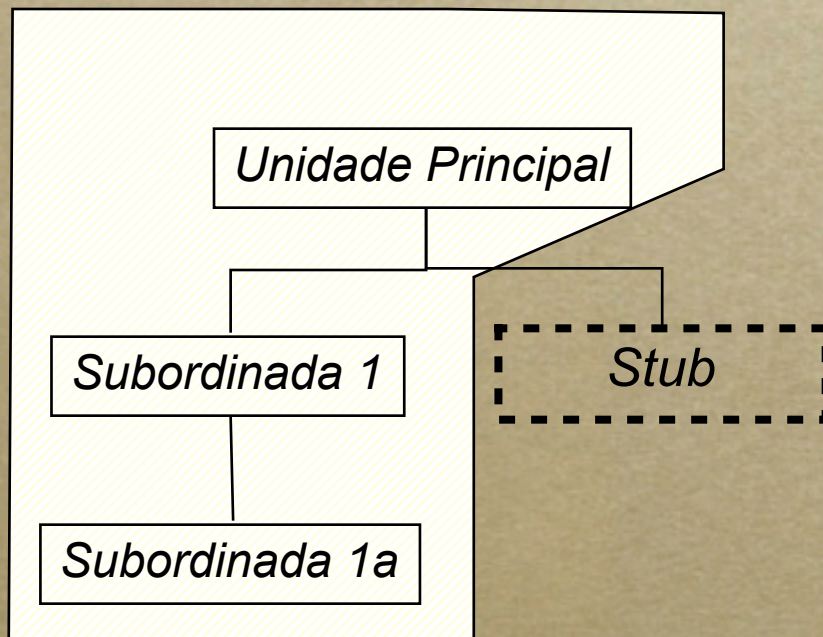
Integração ascendente (“top-down”)

- *começa com a unidade principal e vai aos poucos integrando as unidades subordinadas*
- *em OO: classes de controle primeiro*
- *utiliza stubs em lugar das unidades subordinadas*

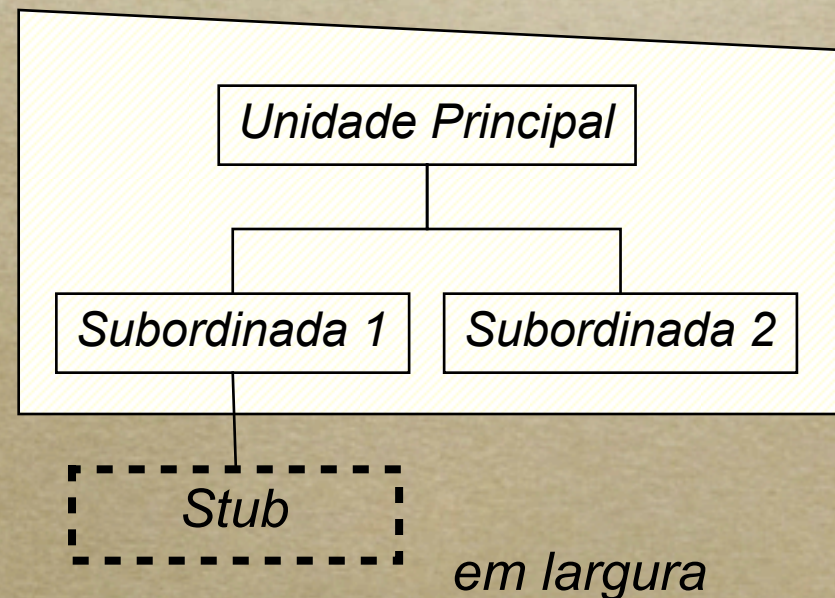


Integração ascendente (“top-down”)

- *Pode ser feita em profundidade ou em largura*
- *interfaces de mais alto nível são testadas mais cedo*



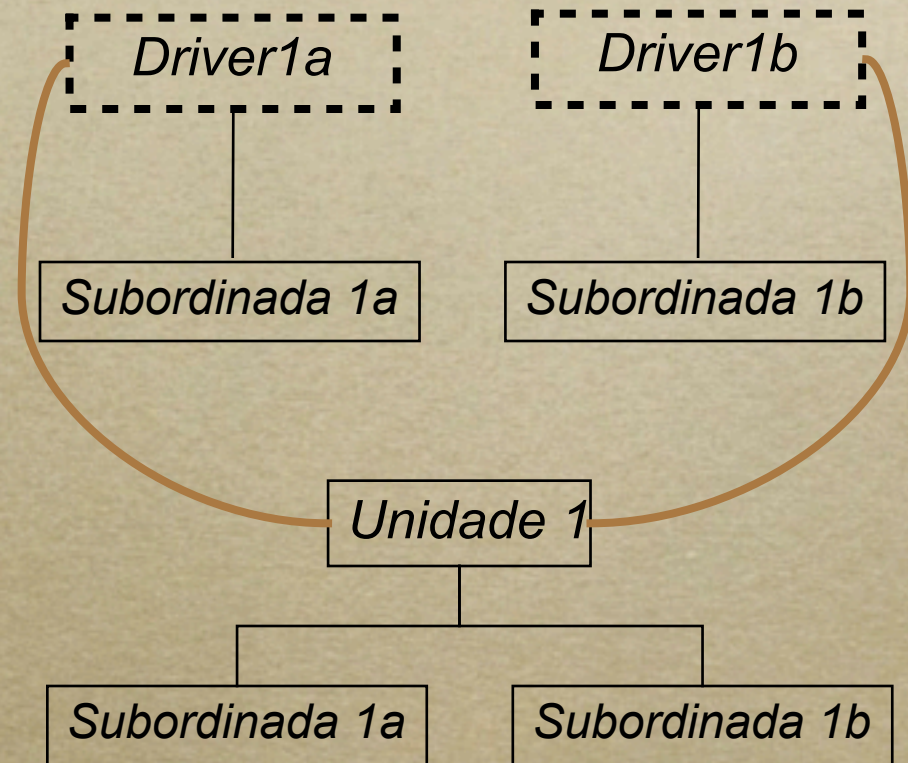
em profundidade



em largura

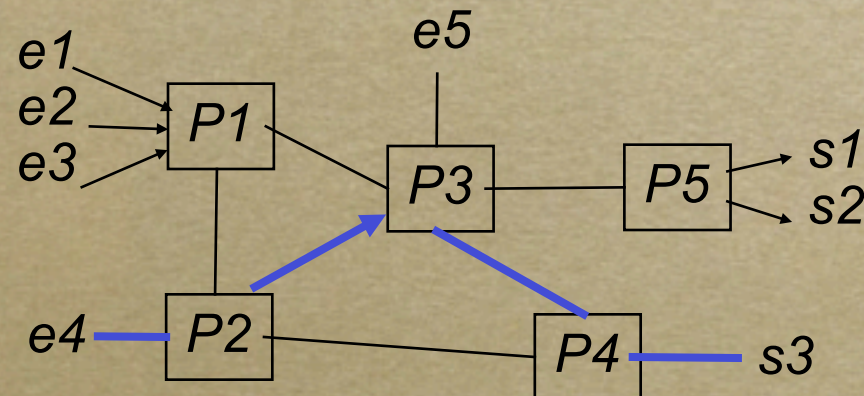
Integração descendente (“bottom-up”)

- *começa a integração pelas unidades subordinadas*
- *em OO: começar pelas classes independentes ou que usam poucas servidoras*
- *utiliza drivers em lugar das unidades de controle*
- *algoritmos de mais baixo nível são testados antes de serem integrados ao resto do sistema*

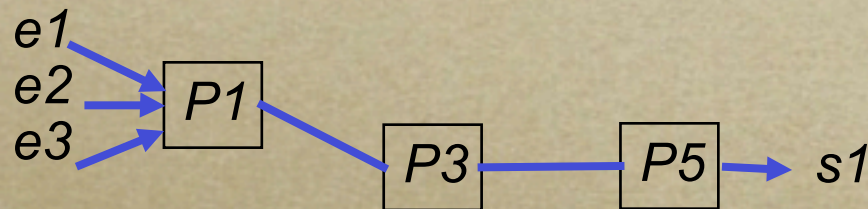
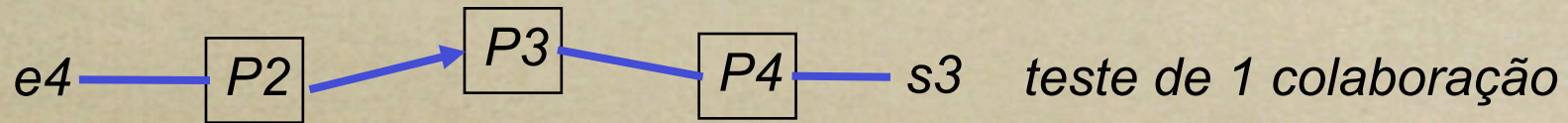


Integração por colaborações

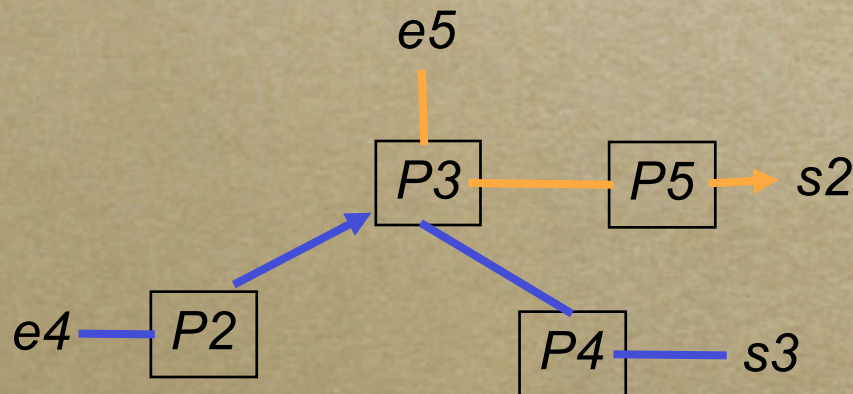
- *identifica colaborações: grupos de unidades que interagem para realizar uma ação do sistema*
- *escolhe uma colaboração integrando as unidades que a compõem*
- *critério: integrar todas as colaborações identificadas*
- *testes enfocam funcionalidades \Rightarrow podem ser reutilizados nos testes de sistema*



Integração por colaborações



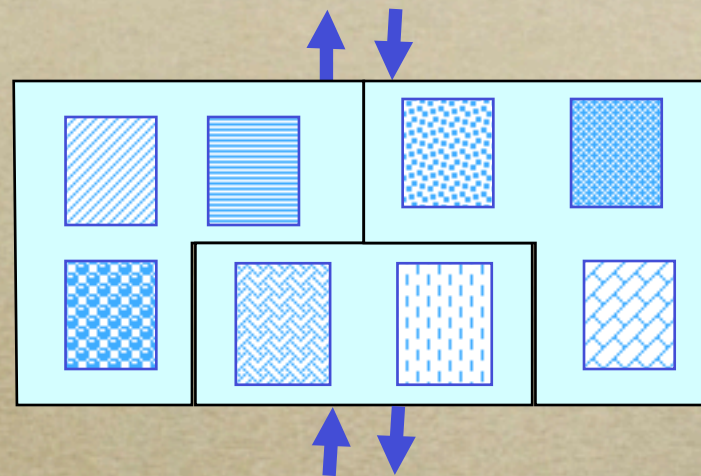
*teste de 1
colaboração com
várias entradas*



*teste de
várias
colaborações*

Testes de sistemas

- *Combinação de testes que tem por objetivo:*
 - *revelar falhas de sistema*
 - *demonstrar que o sistema em teste implementa os requisitos funcionais e não funcionais*
 - *o sistema foi construído corretamente ?*



Fontes de informação para testes

- *especificação de requisitos*
- *protótipo, layouts ou modelos da IU*
- *políticas da organização implementadas como objetos de negócio, “stored procedures” ou “triggers”*
- *características do produto descritas na literatura*
- *características e procedimentos descritos na documentação, telas de ajuda ou assistentes de operação (“wizards”)*

Especificação deve ser:

- **completa**
- **consistente**
- **precisa**
 - **testável**

Testes dos requisitos funcionais

- *Visam verificar se as funcionalidades especificadas foram devidamente implementadas*
- *Uso de métodos de testes caixa-preta :*
 - *partição de equivalência*
 - *valores-limite*
 - *tabela de decisão / grafo causa-efeito*
 - *modelos de estado*
 - *diagramas de casos de uso*
 - *cenários*
 - *...*

Testes dos requisitos não funcionais

- *Visam determinar se a implementação do sistema satisfaz aos requisitos não funcionais*
- *Tipos de testes:*
 - *configuração e compatibilidade*
 - *desempenho*
 - *estresse*
 - *tolerância a falhas*
 - *segurança*
 - *...*

Testes de configuração e compatibilidade

- *Verificam se a implementação é capaz de executar nas diferentes configurações do ambiente alvo que foram especificadas*
- *Verificam se a implementação é capaz de interoperar com sw e hw conforme especificado:*
 - *sistemas já existentes*
 - *plataformas específicas de hw*
 - *diferentes (versões de)sistemas operacionais*
 - *diferentes interfaces gráficas (X-Windows, Microsoft Windows)*
 - *diferentes API e IDL*
 - *diferentes SGBD*

Testes de desempenho

- *Visam determinar se implementação satisfaz aos requisitos de desempenho especificados:*
 - *configuração de rede*
 - *tempo de CPU*
 - *limitação de memória*
 - *carga do sistema*
 - *taxa de chegada de entradas*
 - *esses requisitos devem ser descritos de forma testável*

Exemplo de como expressar requisitos para testes

<i>Desempenho</i>	<ul style="list-style-type: none">• <i>Nro.transações/segundo</i>• <i>tempo de resposta</i>
<i>Tamanho</i>	<ul style="list-style-type: none">• <i>Kbytes</i>• <i>nro. de chips de RAM</i>
<i>Usabilidade</i>	<ul style="list-style-type: none">• <i>tempo de treinamento</i>• <i>número de quadros de ajuda</i>
<i>Confiabilidade</i>	<ul style="list-style-type: none">• <i>tempo médio entre defeitos (failure)</i>• <i>taxa de ocorrência de defeitos</i>• <i>disponibilidade</i>• <i>tempo de reinicialização após defeito</i>• <i>% de eventos que levam a defeitos</i>• <i>probabilidade de que dados sejam corrompidos</i>
<i>Portabilidade</i>	<ul style="list-style-type: none">• <i>% comandos dependentes de plataforma</i>• <i>número de plataformas-alvo</i>

Variações dos testes de desempenho

- *Testes de carga*
 - *geralmente associados com sistemas transacionais*
 - *usam simuladores de carga para geração de múltiplas transações/usuários simultâneamente*
- *Testes de volume*
 - *geralmente usados para sistemas “batch”*
 - *consistem na transmissão de um grande volume de informações quando o sistema está com a carga normal*

Testes de estresse

- *Visam verificar se o sistema não apresenta um comportamento de risco quando submetido a carga elevada e com um ou mais recursos saturados*
- *Importância:*
 - *muitos sistemas apresentam comportamento de risco nessa situação*
 - *falhas detectadas são sutis*
 - *correções desse tipo de falha podem requerer retrabalho considerável*

Testes da tolerância a falhas

- *Visam verificar os mecanismos de detecção e recuperação de erros:*
 - *recuperação automática:*
 - *mecanismos implementados corretamente ?*
 - *tempo de resposta é aceitável ?*
 - *Recuperação manual*
 - *o tempo médio de reparo é aceitável ?*
- *São realizados usando-se testes por injeção de falhas*

Testes de segurança

- *Visam verificar a capacidade do sistema de impedir acesso não autorizado, sabotagem ou outros ataques intencionais*
- *Características básicas de segurança são testadas como as outras funcionalidades (logon/logoff, permissões)*
- *testes adicionais são geralmente feitos por especialistas ou “hackers” contratados*

Testes de validação

- *Têm os mesmos objetivos que os testes de sistemas, só que envolvem a participação do cliente ou usuário*
- *Testes alfa:*
 - *realizados por um grupo de usuários no ambiente de desenvolvimento*
 - *seu objetivo é determinar se o sistema pode ser liberado*
- *Testes beta*
 - *realizados por um grupo de usuários em ambiente de operação*
- *Testes de aceitação*

Testes de validação

- *Testes de aceitação:*
 - *realizados pelo cliente para decidir se aceita ou não o sistema*
 - *são realizados de acordo com um plano de aceitação*
- *Testes de conformidade:*
 - *visam verificar se a implementação satisfaz à legislação ou a padrões estabelecidos*
 - *podem ser realizados por centros de certificação*

Testes de regressão

- *Testes realizados a cada vez que um sw é alterado*
- *Objetivo:*
 - *validar modificações feitas*
 - *mostrar que modificações realizadas não afetaram as partes não modificadas*
 - *isto é*
 - *mostrar que o sw não regrediu*

Aplicabilidade dos testes de regressão

- *testar aplicações críticas que devem ser retestadas frequentemente*
- *testar sw que é alterado constantemente durante o desenvolvimento (por exemplo, Processo Incremental)*
- *testar componentes reutilizáveis para determinar se são adequados para o novo sistema*
- *durante os testes de integração*
- *durante os testes, após correções*
- *em fase de manutenção*
- *para identificar diferenças no comportamento do sistema quando há mudanças de plataforma (uso de seqüências-padrão ou “benchmarks”)*

Aplicabilidade dos testes de regressão (OO)

- *quando uma nova subclasse é criada*
- *quando uma super-classe é alterada*
- *quando uma classe servidora é alterada*
- *quando uma classe é reusada em um novo contexto*

Abordagens para os testes de regressão

- *Retesta tudo:*
 - *reaplica todos os testes*
- *Seletiva:*
 - *seleciona um subconjunto dos testes aplicados*
- *Em ambos os casos:*
 - *alguns testes devem ser atualizados ou novos testes devem ser acrescentados*

Sumário

- *Testes de integração são úteis mesmo que as unidades tenham sido testadas*
- *Abordagens de integração incremental são “mais baratas” do que abordagem não-incremental*
- *Testes de desempenho, estresse e tolerância a falhas devem ser realizados cedo na fase de testes pois podem implicar em grandes alterações (talvez de projeto)*
- *Modificações no sw são inevitáveis e introduzem falhas ⇒ testes aplicados precisam ser armazenados para uso em testes de regressão*