

Projeto e Desenvolvimento de Software



Prof. Dr. Ronaldo C. de Oliveira
ronaldo.co@ufu.br
FACOM - 2019

Verificação e Validação (V&V)

S.L.Pfleeger (Cap.8 & 9)

R.Pressman (Cap.13 & 14)

I.Sommerville (Cap.22 & 23)

Introdução

- Verificação e Validação
 - Atividade da Eng.Software na qual procura-se detectar a **correção** do software (que está sendo) produzido.
 - Correção em que sentido?

Introdução

- Tipos de Correção
 - O produto atende sua especificação?
 - **Verificação**
 - *Nos fizemos as coisas corretamente?*
 - O produto atende as necessidades do usuário?
 - **Validação**
 - *Nos fizemos as coisas de acordo com o esperado?*

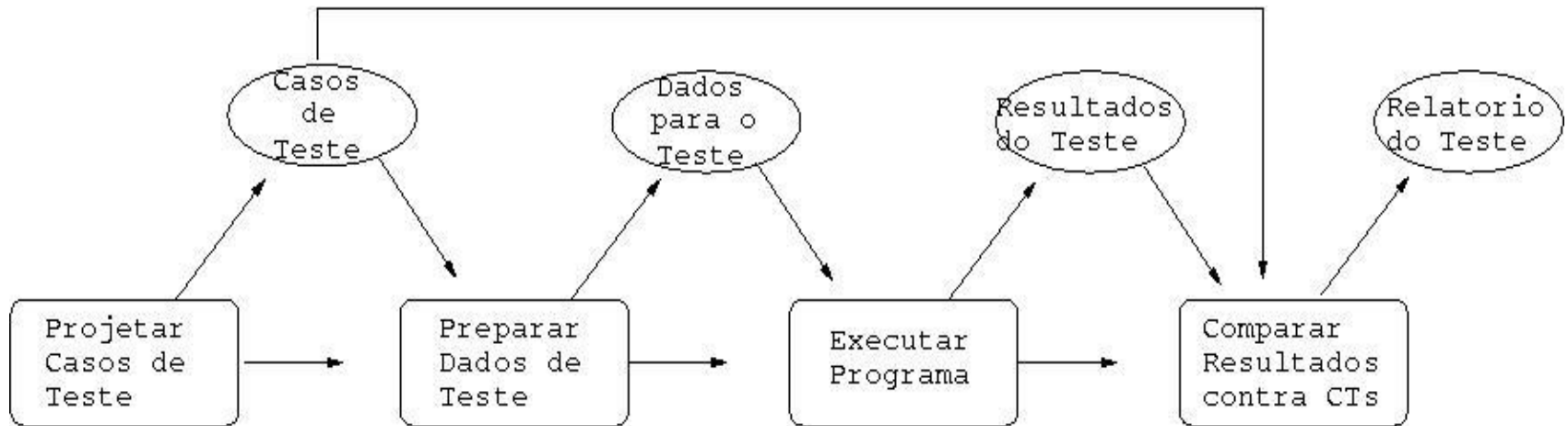
Teste

- Trata-se da **verificação** de um programa.
 - O programa atende sua especificação?
- Quem deve testar um programa?
 - Alguém que **não** seja programador ou autor do mesmo.

Teste

- O que esperar de um teste?
 - **Erro!**
- Qual é o **produto** de um teste?
 - Um relatório com os resultados do teste.
- Vejamos o processo.

Teste: o processo



ref.:I.Sommerville



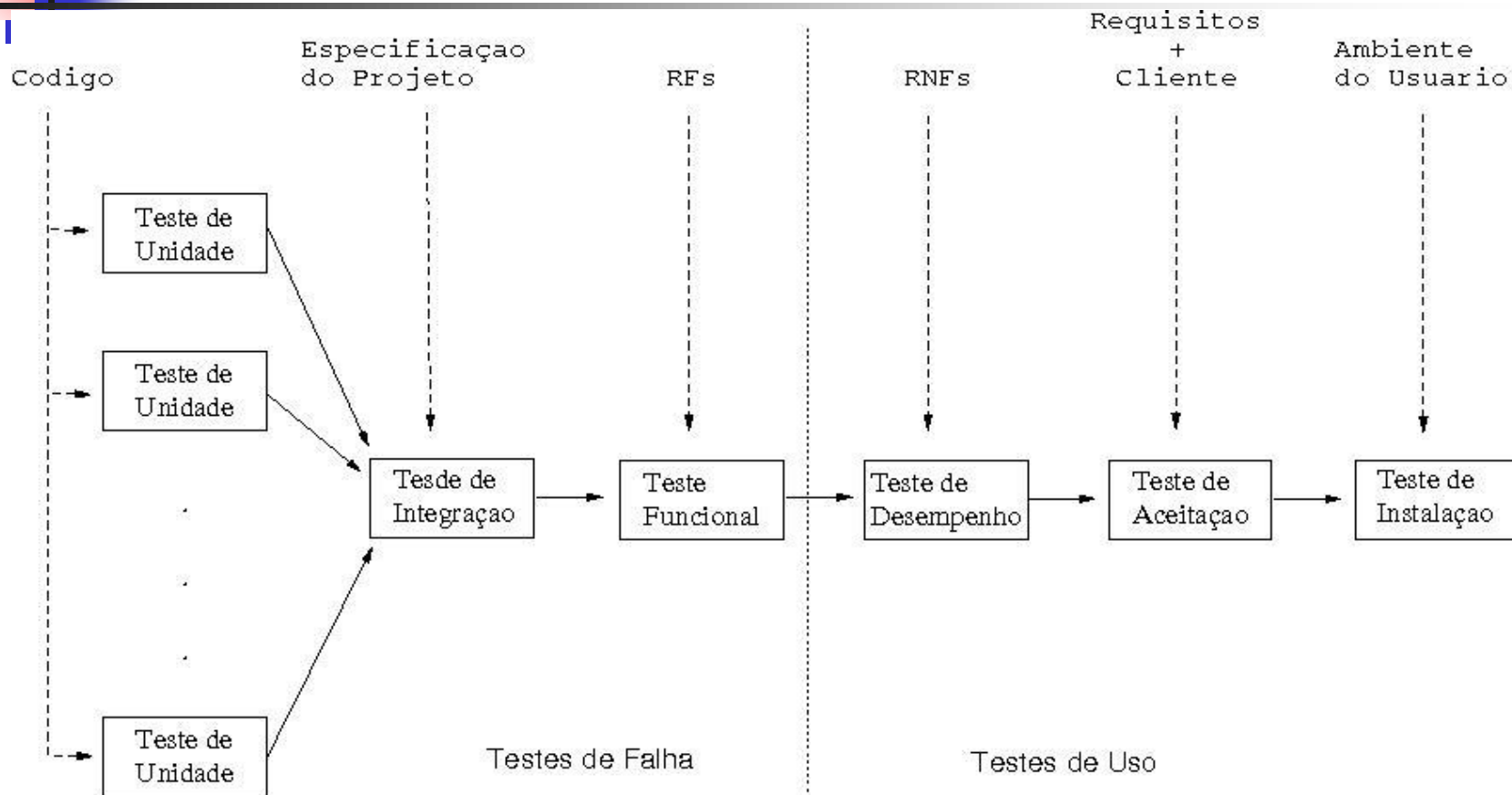
Um Teste

- É um conjunto de **casos de teste**.
- O que é um Caso de Teste (CT)?
 - É uma escolha específica de um dado que irá servir para testar um programa.
 - Estabeleça objetivos para o CT.

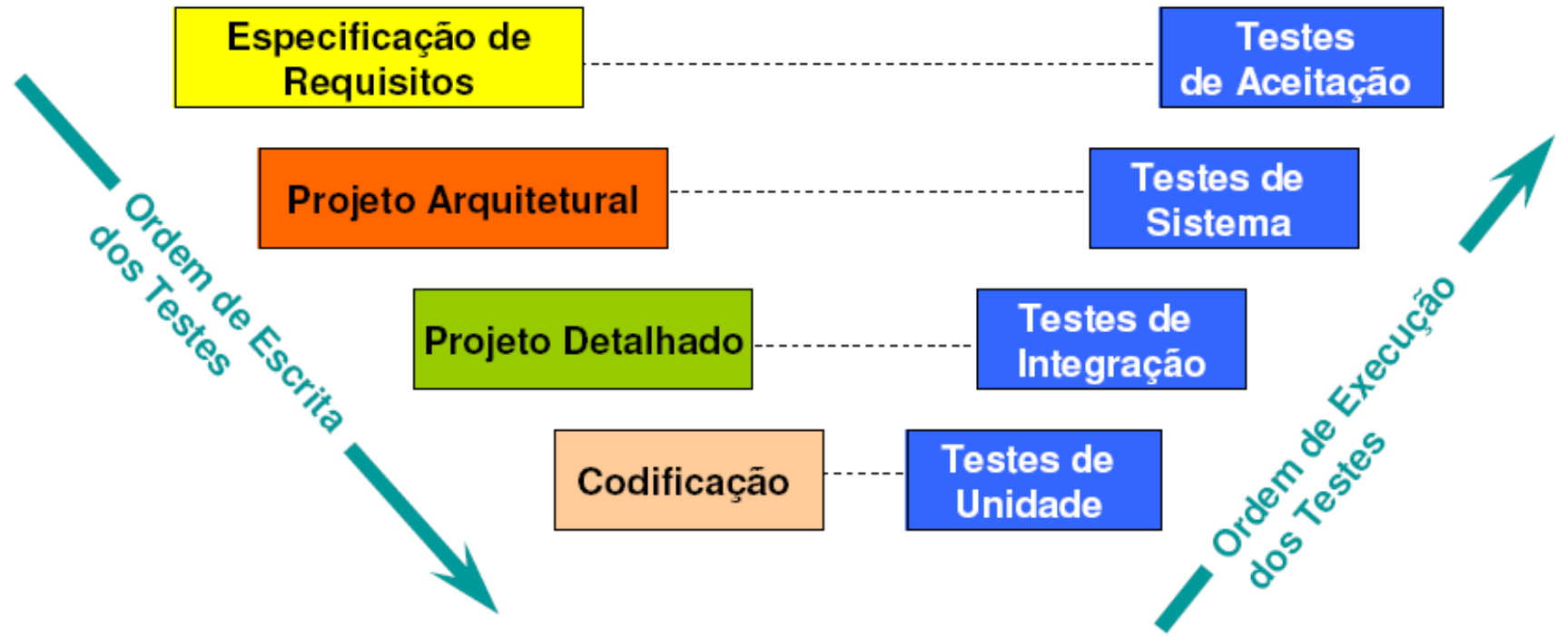
Tipos de Teste

- Dois tipos:
 - Falha
 - Uso
 - Visa determinar o comportamento do programa em condições normais de uso
 - Desempenho
 - Confiabilidade
- Os testes são realizados em diferentes etapas.

Etapas de Teste



Etapas de Teste



Teste de Unidade

- Teste de Unidade é aquele que realizamos em programas.
- Há diferentes abordagens:
 - inspeção
 - caixa preta
 - caixa branca
 - caminho
 - ...



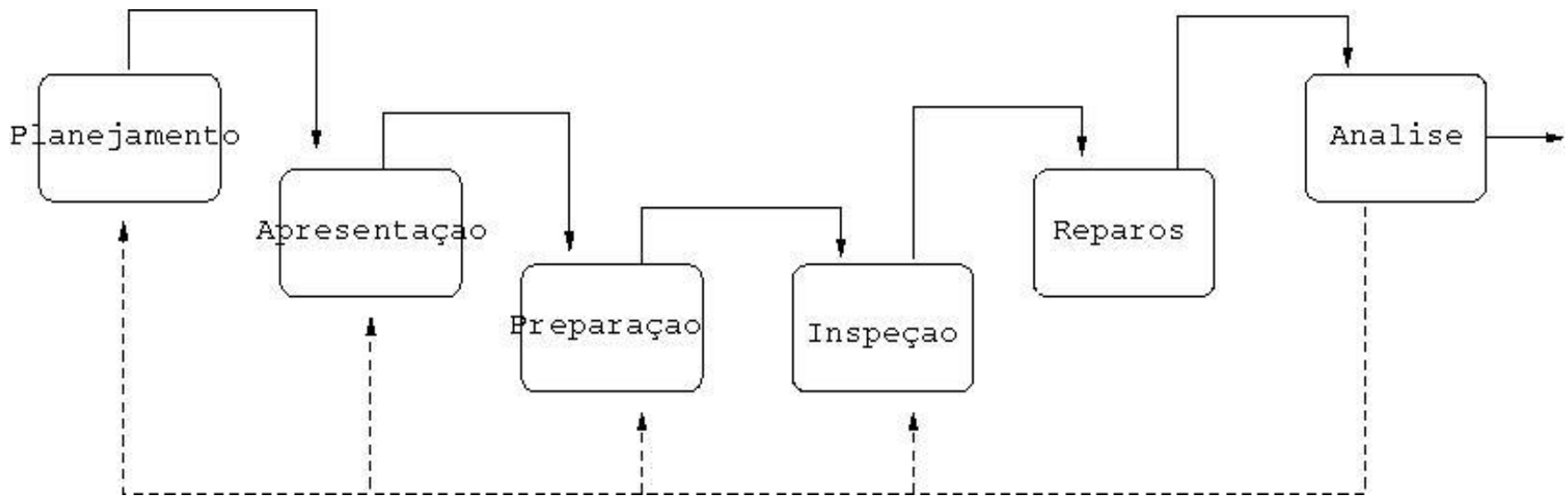
Teste de Unidade: Inspeção

- Trata-se do exame do código-fonte antes do mesmo ser executado.
- O exame se dá por meio da **leitura** do código por parte de algumas pessoas envolvidas com o sistema.

Teste de Unidade: Inspeção

- Inspecionar um código, em geral, envolve diversas reuniões.
- Os participantes de uma inspeção:
 - moderador,
 - autor,
 - programadores experientes.

Teste de Unidade: Inspeção



ref.:I.Sommerville



Teste de Unidade: Inspeção

- A inspeção é conduzida por um moderador.
- Uma inspeção pode demandar até “duas horas”.
- Pode-se detectar:
 - erros potenciais / lógica,
 - eficiência de algoritmo e de uso de estrutura de dados,
 - obediência a padrões.

Teste de Unidade: Inspeção

- Que erros são encontrados?
 - Erros de dados
 - *Iniciação de variáveis; min/max de arrays*
 - Erros de Controle
 - *condição correta? laço termina?*
 - Entrada/saída
 - *formato; estrutura de dados/registo*
 - Gerência de Memória
 - *uso de ponteiros: lista encadeada*



Teste de Unidade: Inspeção

- É um processo **coletivo** de detecção de erro.
- À medida em que é utilizado, a organização pode detectar **padrões de erros**.
- **Processo efetivo** para detecção de erros.
- Dificuldade de adoção.



Teste de Unidade: Caixa Preta

- Procura verificar se o programa atinge seu(s) objetivo(s).
- Os Casos de Testes são construídos com base:
 - RF(s);
 - entrada \Rightarrow saída;
 - partições.

Teste de Unidade: Caixa Preta

- Teste partições
 - Trata-se do teste com determinados valores de dados que representam **partições de equivalência** quanto ao teste.
 - Exemplo:
 - Fila
 - primeiro, meio, último
 - Pesquisa binária
 - menor, maior, menor < u < maior

Teste de Unidade: Caixa Preta

- Procura detectar:
 - funções incorretas ou omissas;
 - erros de interface;
 - erros de estrutura de dados ou de acesso a base de dados externa;
 - erros de comportamento ou **desempenho**;
 - erros de iniciação ou término.



Teste de Unidade: Caixa Preta

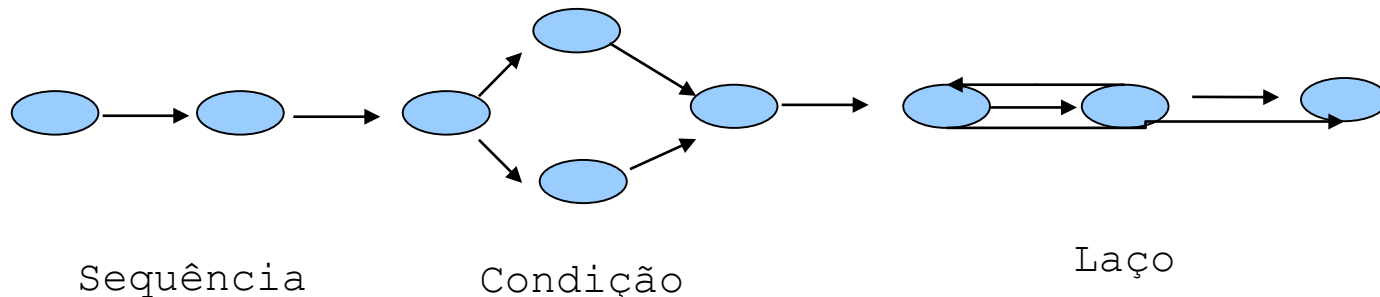
- O teste de Caixa Preta foca no comportamento da unidade.
- Não é baseado no conhecimento de como a unidade foi implementada.
- Não tem como objetivo analisar o desempenho do programa.

Teste de Unidade: Cx. Branca

- O Caso e Teste é construído com base no conhecimento do código da unidade.
- O testador lê o código e prepara um Caso de Teste que visa forçar a execução de um trecho do programa.
- Complementa o Teste de Caixa Preta.

Teste de Unidade: Caminho

- É um tipo de Teste de Caixa Branca.
- Um grafo de caminhos do programa é derivado da lógica do mesmo.
- Estruturas que compõem o grafo:



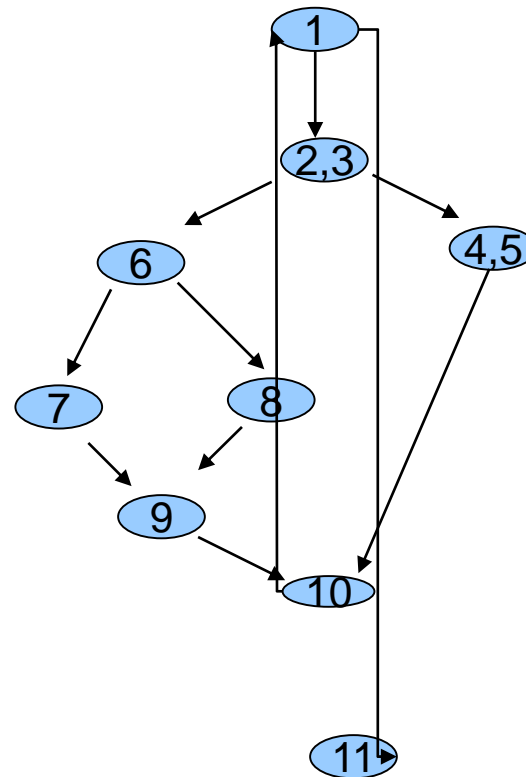


Teste de Unidade: Caminho

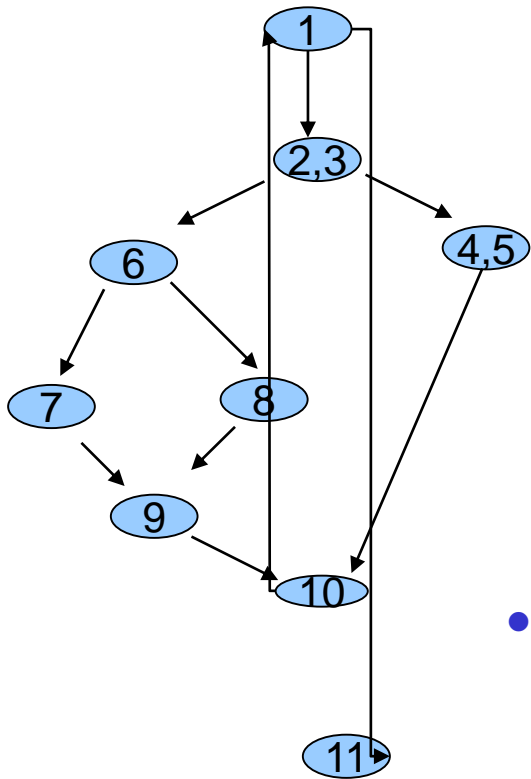
- Com base no grafo de caminhos, é possível determinar os **caminhos independentes** do programa.
- Qual é o grafo do programa a seguir?

Teste de Unidade: Caminho

- 0 f() {
- 1 while (condA) {
- 2 atribA;
- 3 if (condB){
- 4 atribB;
- 5 atribC;
- else
- 6 if (condC)
- 7 atribD;
- else
- 8 atribE;
- 9 }
- 10 }
- 11 }



Teste de Unidade: Caminho



Quais são os caminhos independentes?

- 1-11
- 1-2-3-4-5-10-1-11
- 1-2-3-6-8-9-10-1-11
- 1-2-3-6-7-9-10-1-11

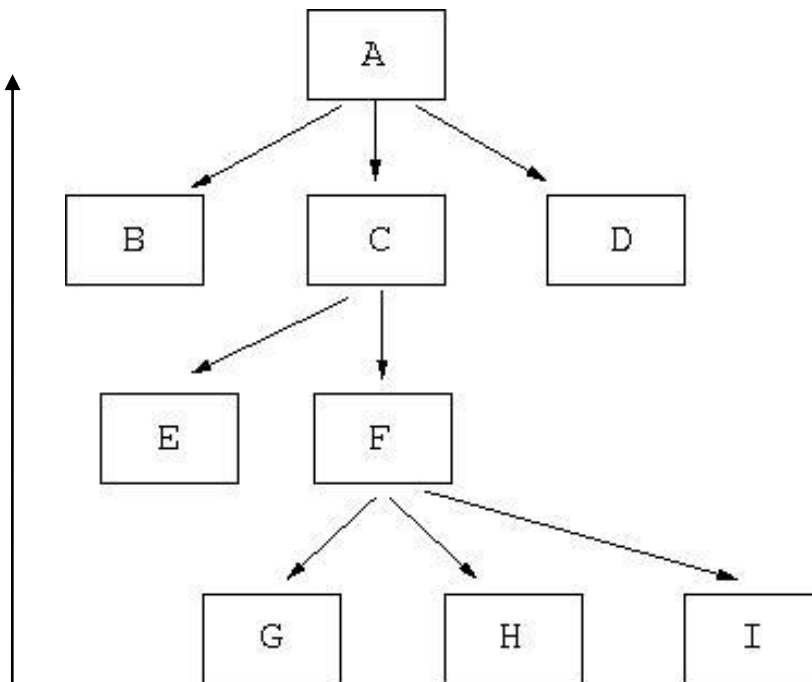
- Complementa o teste de caixa preta.
- Considerar a intensidade de V&V.

Teste de Integração

- Trata-se do teste de várias Unidades “trabalhando” em conjunto.
- Qual é o principal erro que procura se detectar?
 - Erro de interface.
- Há diferentes estratégias.

Teste de Integração: Estratégias

- Botton-UP

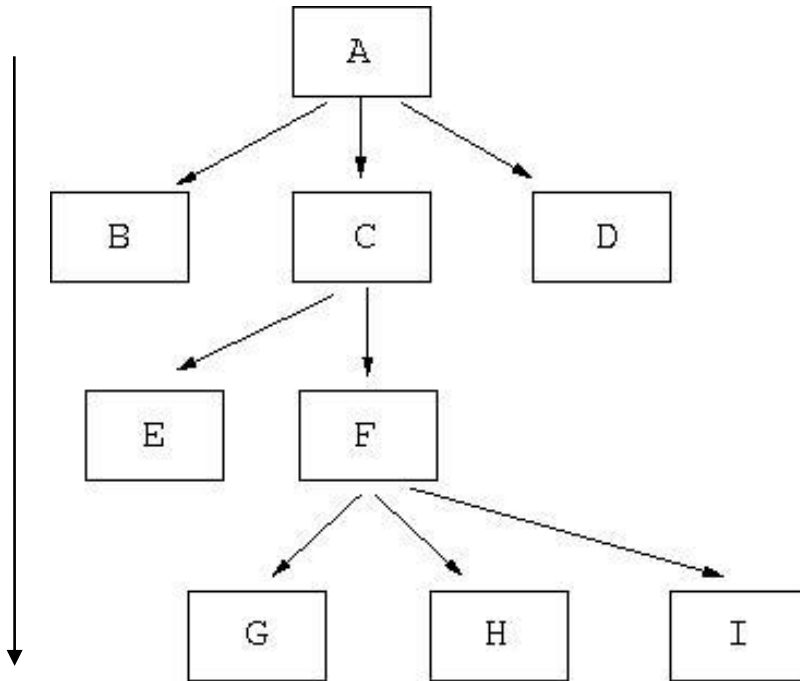


As unidades necessitam estar finalizadas.

As unidades são integradas (uma a uma ou em pequenos grupos) .

Teste de Integração: Estratégias

- Top-Down



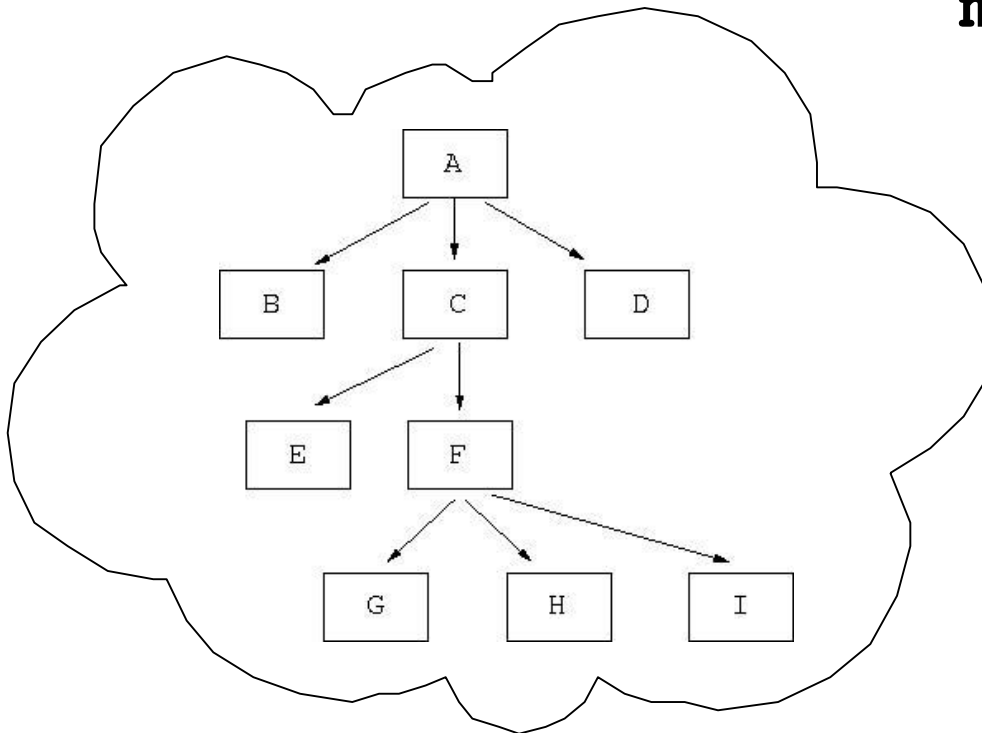
As unidades não precisam estar finalizadas.

Necessidade de códigos que substituam o que será ainda desenvolvido (**stubs**).

Teste de Integração: Estratégias

- Big-Bang

Junta tudo ao mesmo tempo.

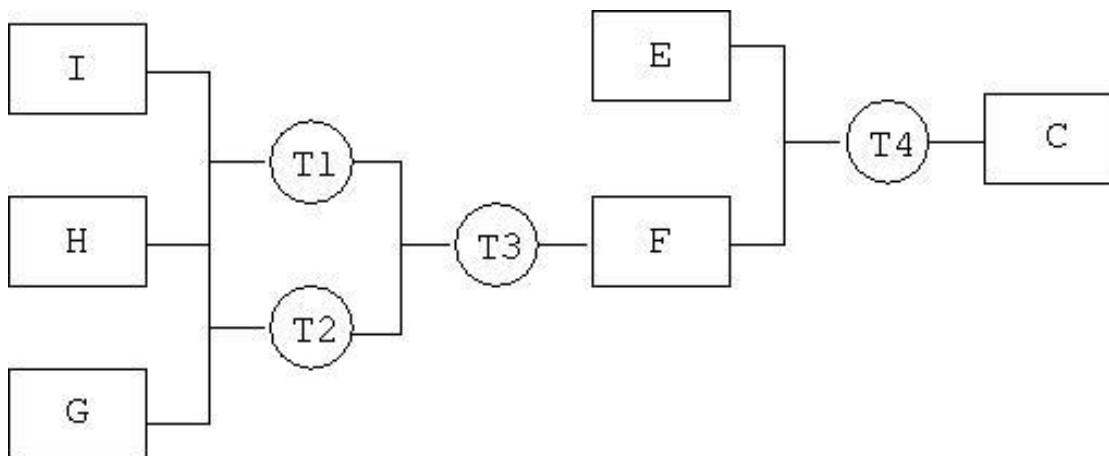


Teste de Integração: Considerações

- Botton-UP
 - Retarda os testes de componentes “importantes”.
- Top-Down
 - Necessidade de **stubs**.
- Big-Bang
 - Utilizado quando o sistema tiver “baixa” complexidade.

Teste de Integração: Abordagem

- Incremental
 - Ampliar os testes a cada componente integrado.
 - Considere os **Teste de Regressão**.





Teste de Integração: Regressão

- Teste de Regressão
 - Aplicar os testes realizados anteriormente à integração de um novo componente.
 - Por quê?
 - O novo componente pode apontar erro nos componentes testados anteriormente.
 - O novo componente pode provocar um novo erro.



Teste de Sistema

- Os elementos de software são integrados com o ambiente operacional (hardware, pessoas, etc.) e testados como um todo.
- Tem o objetivo de verificar o funcionamento da aplicação em um ambiente operacional similar ao de produção.
- O QUE É TESTADO:
 - As funcionalidades definidas na especificação.



Teste de Sistema

- Tipos de Teste de Sistema:
 - **Teste funcional:**
 - Testa a funcionalidade geral (regras de negócio) bem como as condições válidas e inválidas .
 - **Teste de recuperação de falhas:**
 - O software é forçado a falhar de diversas maneiras verificando a eficiência dos procedimentos de recuperação.

Teste de Sistema

- Tipos de Teste de Sistema:
 - **Teste de desempenho:**
 - Verifica o tempo de resposta e processamento para diferentes configurações:
 - número de usuários ou
 - tamanho do BD.
 - Geralmente ocorre paralelamente ao teste de estresse.
 - **Teste de segurança e controle de acesso:**
 - Verifica se todos os mecanismos de proteção de acesso e de dados estão funcionando.

Teste de Sistema

- Tipos de Teste de Sistema:
 - **Teste de interfaces com o usuário:**
 - Verifica navegação, consistência e aderência a padrões
 - **Teste de volume (ou carga):**
 - Verifica se o sistema suporta altos volumes de dados.



Teste de Aceitação

- O software é testado pelo usuário final
- Tem como objetivo demonstrar a conformidade com os requisitos definidos pelo usuário
- O QUE É TESTADO:
 - Funcionalidades testadas seguindo a abordagem “caixa preta”.

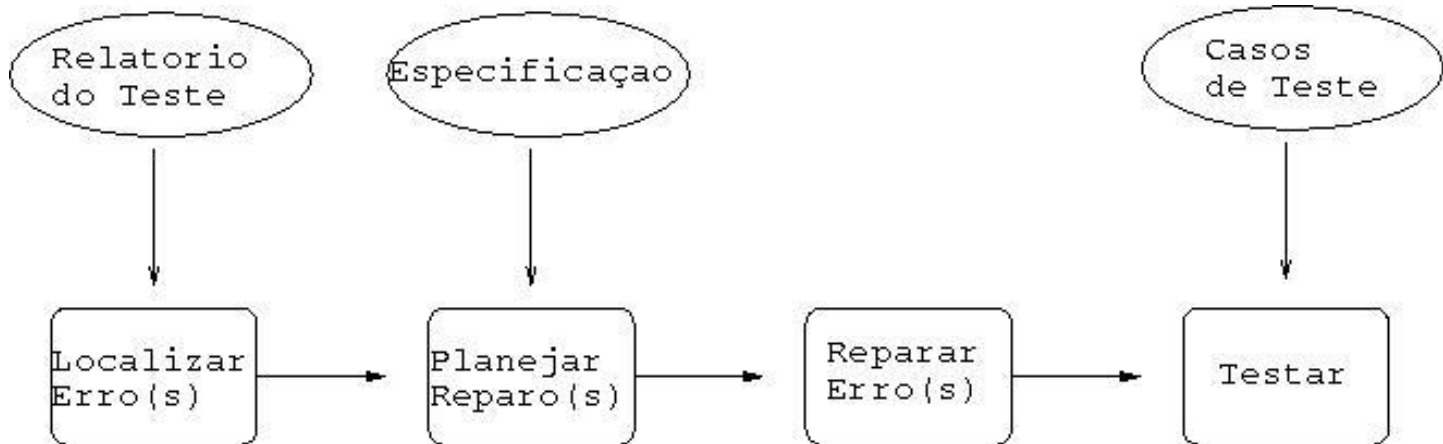


Teste de Aceitação

- Teste de Aceitação podem ser de dois tipos:
 - Testes alfa:
 - teste feito pelo usuário, geralmente nas instalações do desenvolvedor em ambiente controlado;
 - desenvolvedor observa e registra falhas ou inconsistências.
 - Testes beta:
 - teste feito pelo usuário, geralmente em suas próprias instalações, em ambiente não controlado
 - sem a supervisão do desenvolvedor
 - Os erros detectados pelos usuários são reportados ao desenvolvedor para correção.

Depuração

- É o processo de encontrar e corrigir os erros encontrados nos testes.
- Como realizar a depuração de um programa?



Critério de Cobertura (ou critério de completude)

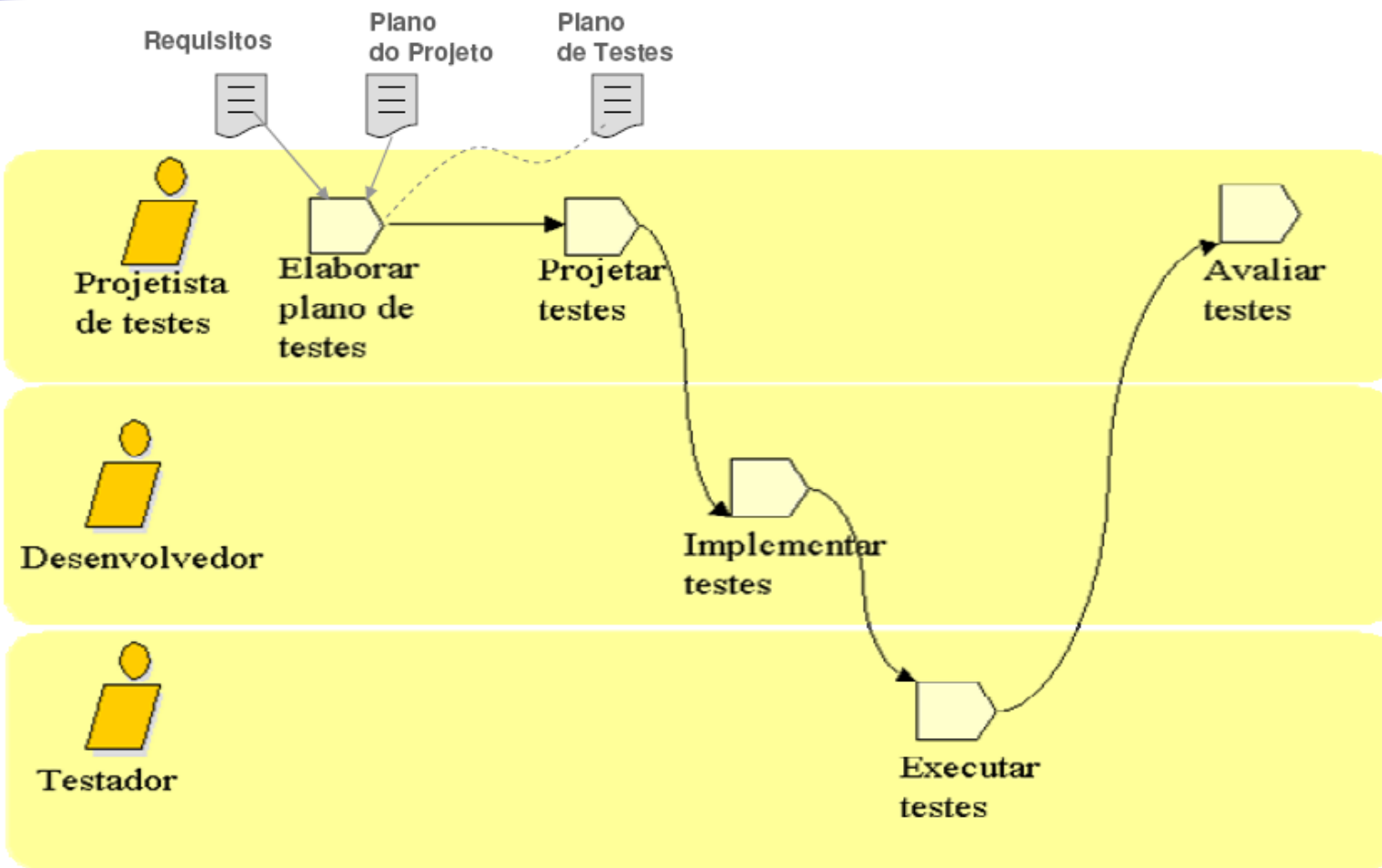
- Quando parar de testar?
- Como saber se o meu conjunto de testes é suficiente?
- O critério de cobertura nos permite avaliar uma bateria de testes.
- Ex:
 - Cobertura de linha:
 - Este critério requer que todas as linhas do programa sejam executadas ao menos uma vez durante os testes;
 - Cobertura de aresta:
 - Este critério requer que todos os caminhos lógicos do programa sejam percorridos durante os testes;



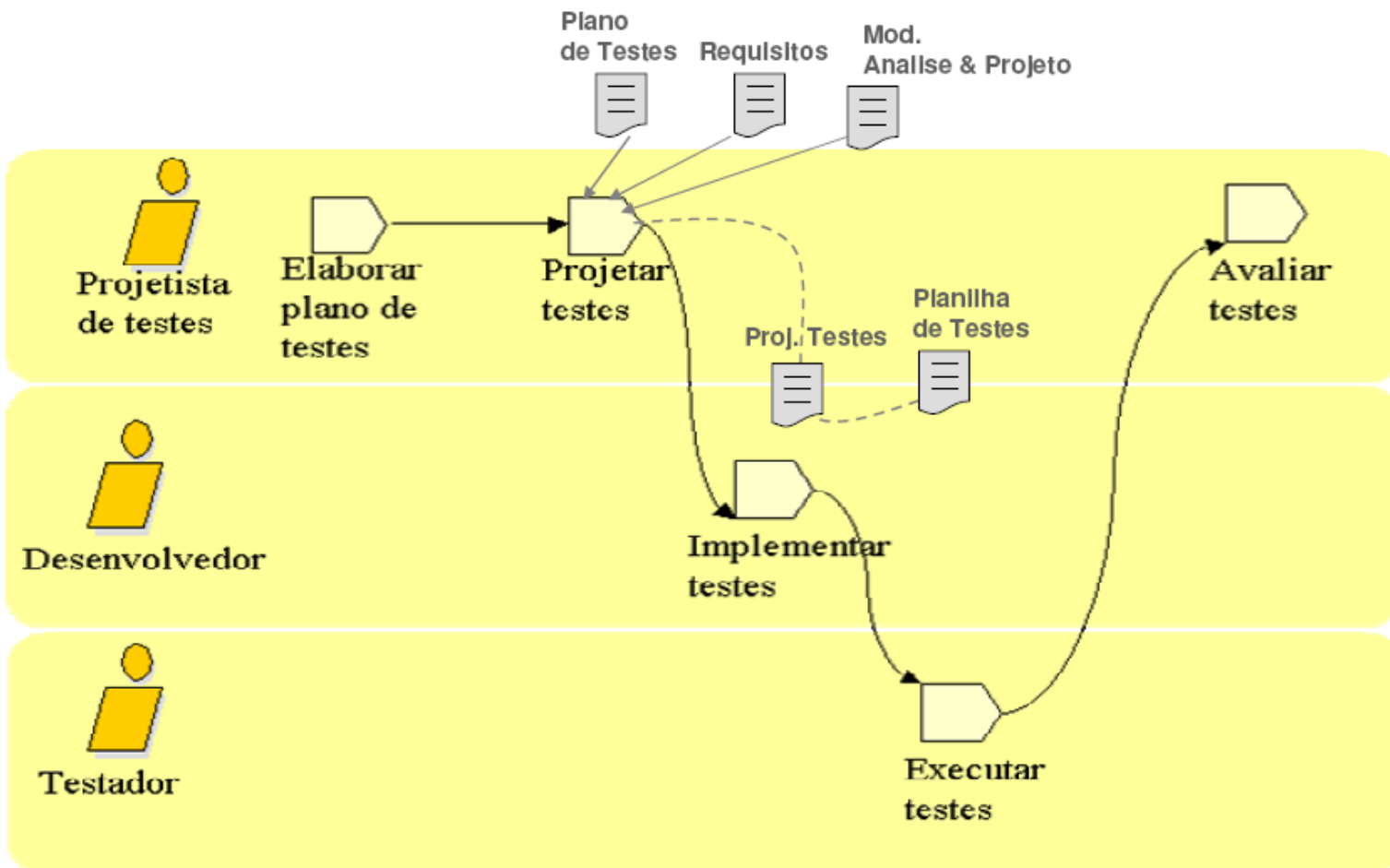
Processo de Testes

- Testes são escritos e gerados ao longo do desenvolvimento.
- Precisamos definir:
 - Quais atividades devem ser seguidas a fim de que o software seja testado?
 - Quando essas atividades serão executadas?
 - Quem são os responsáveis por cada atividade de teste?
- Isto é o que define um processo de testes.

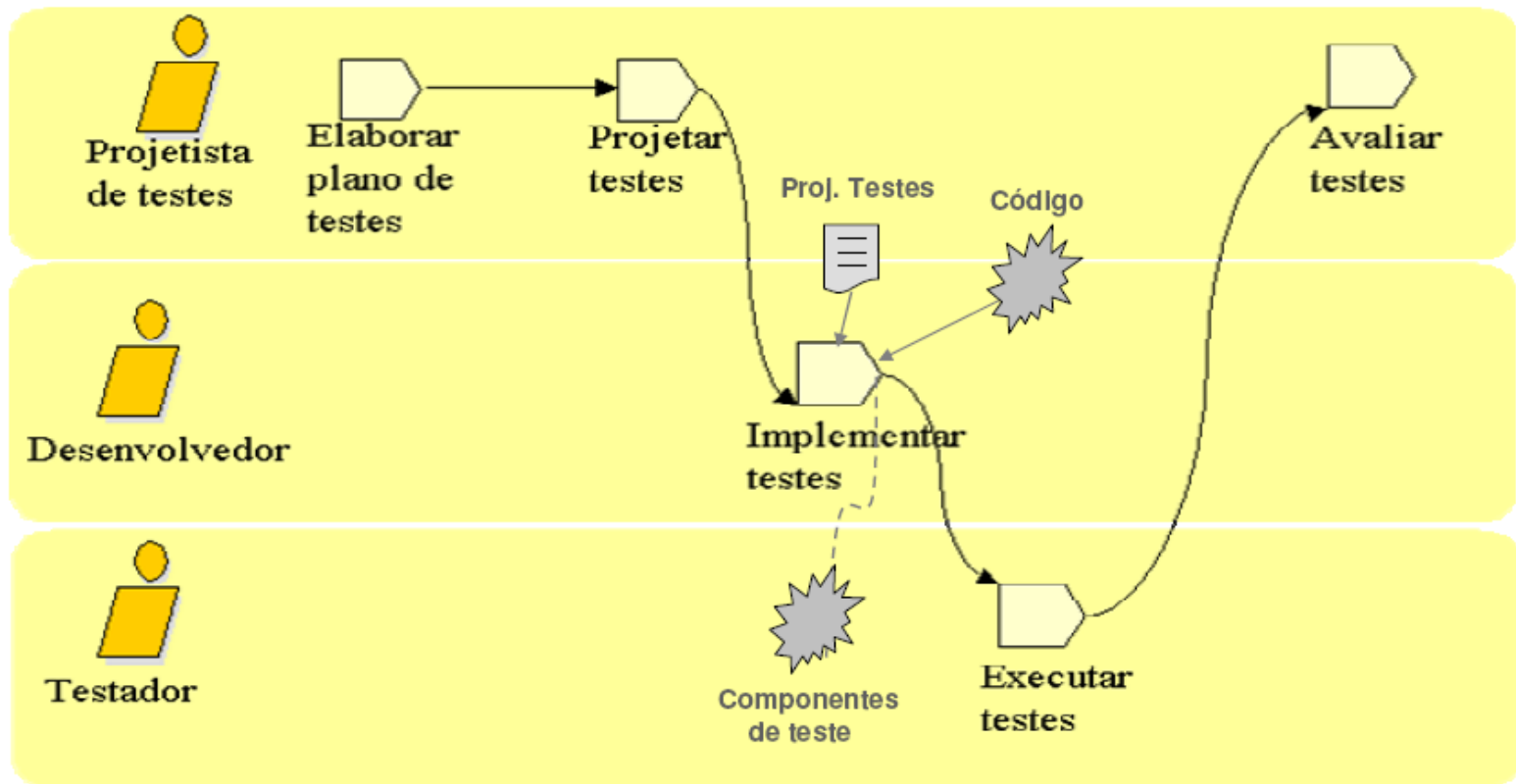
Processo de Testes: Exemplo



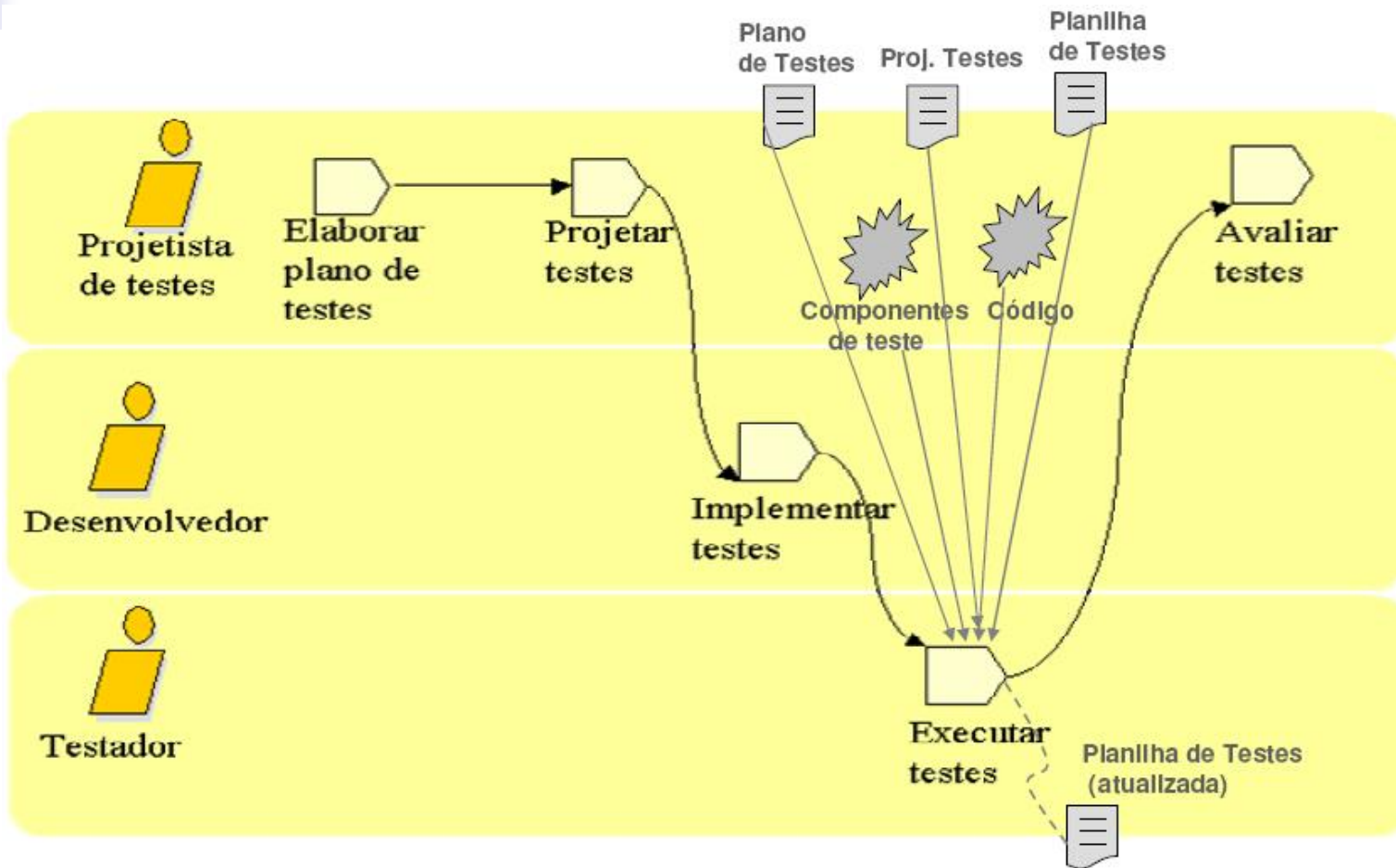
Processo de Testes: Exemplo



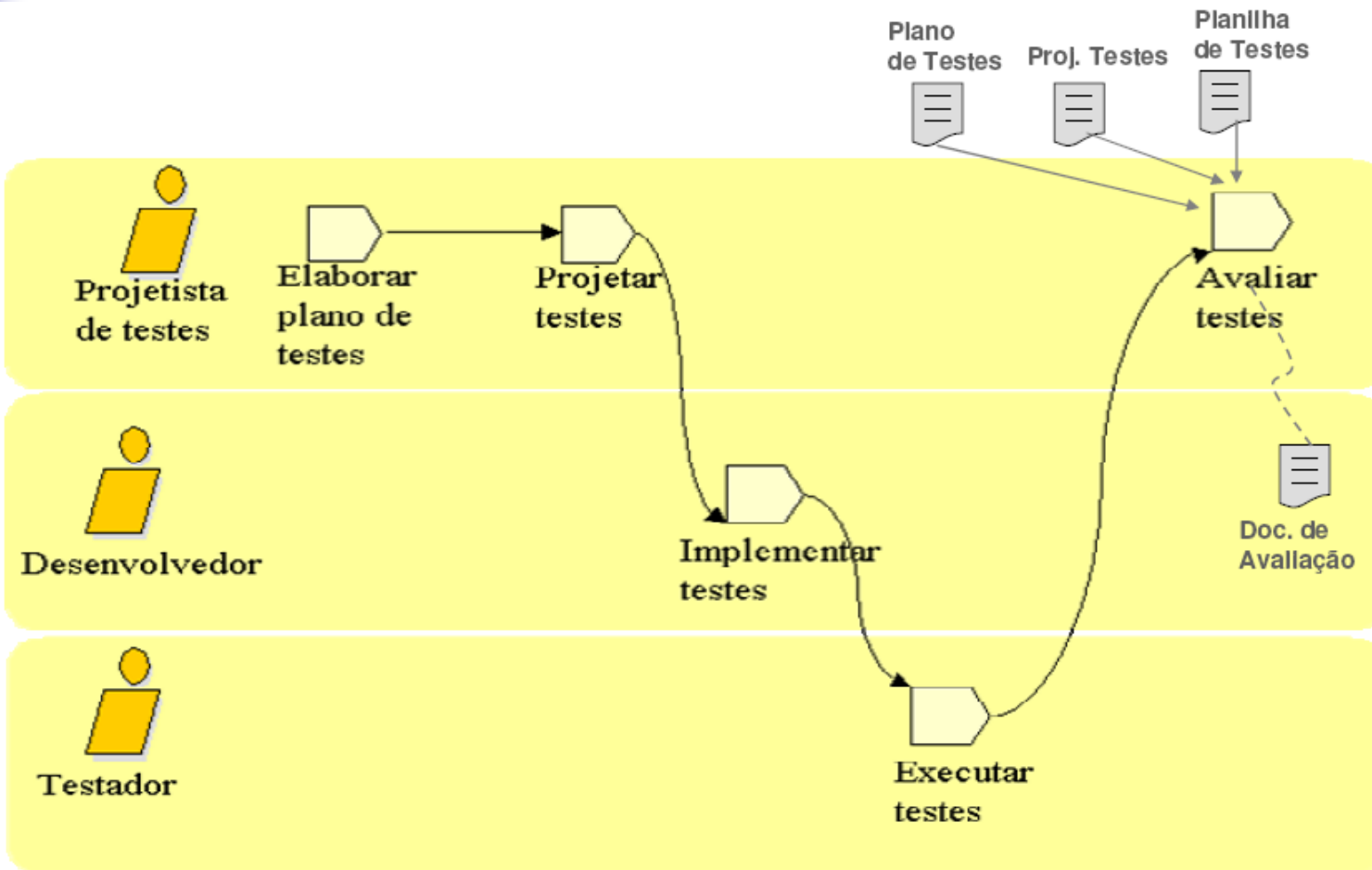
Processo de Testes: Exemplo



Processo de Testes: Exemplo



Processo de Testes: Exemplo





Processo de Testes: Exemplo

- 1 - Planejamento de testes:
 - São definidos os requisitos a serem testados, e ocorre uma priorização dos requisitos selecionados.
 - São estimados o cronograma das atividades de teste, o esforço e os recursos exigidos.
- 2 - Projeto dos testes:
 - Os casos e os procedimentos de teste são escritos.
- 3 - Implementação dos testes:
 - São desenvolvidos os componentes de teste.
- 4 - Execução dos testes:
 - Pode ser realizada de forma manual e/ou automática.
- 5 - Avaliação dos testes:
 - São verificados a cobertura dos testes, a tendência dos defeitos e se os critérios de sucesso foram atingidos.