Lição 6

Testes de Software

Objetivos

Ao final desta lição, o estudante será capaz de:

- Utilizar métodos e técnicas no desenvolvimento de testes de casos
- Utilizar métodos do projeto de casos de teste
- Aprender a como testar classes
- Entender o desenvolvimento voltado para testes
- Aprender a testar sistemas
- Utilizar testes de métricas

Teste de Software

- Identificar as falhas e erros nos sistemas
- Criar testes que tenham grande eficácia na busca por erros
- Executar o programa com o intuito de encontrar erros
- Considerado eficaz quando erros são detectados ou corrigidos

Falhas

- Identificação de Falhas
- Correção ou Remoção da Falha

Princípios no Teste de Software

- Testes são baseados nos requerimentos
- Testes são planejados antes de serem iniciados
- Testes aplicam o Princípio de Pareto
- Testes devem começar “pequenos” e sendo gradativamente implementados
- Testes são exaustivos de forma a desenvolver um sistema com uma menor chance de conter falhas
- Testes são conduzidos por um terceiro e independente do grupo

Estratégia de Teste de Software

- Verificação e Validação
- Desenvolvedores e o Grupo de Garantia de Qualidade
- Especificação de Teste

Teste Caixa-Branca

- Conhecido também como teste da caixa de vidro
- Técnica que utiliza as estruturas de controles internas
- Assegurar que as operações internas ocorreram como o especificado
- Assegurar que nenhum erro lógico, instruções incorretas e erros de tipografia serão esquecidos

Teste Básico do Trajeto

- Permite que se crie testes complexos baseados na especificação procedural de um componente

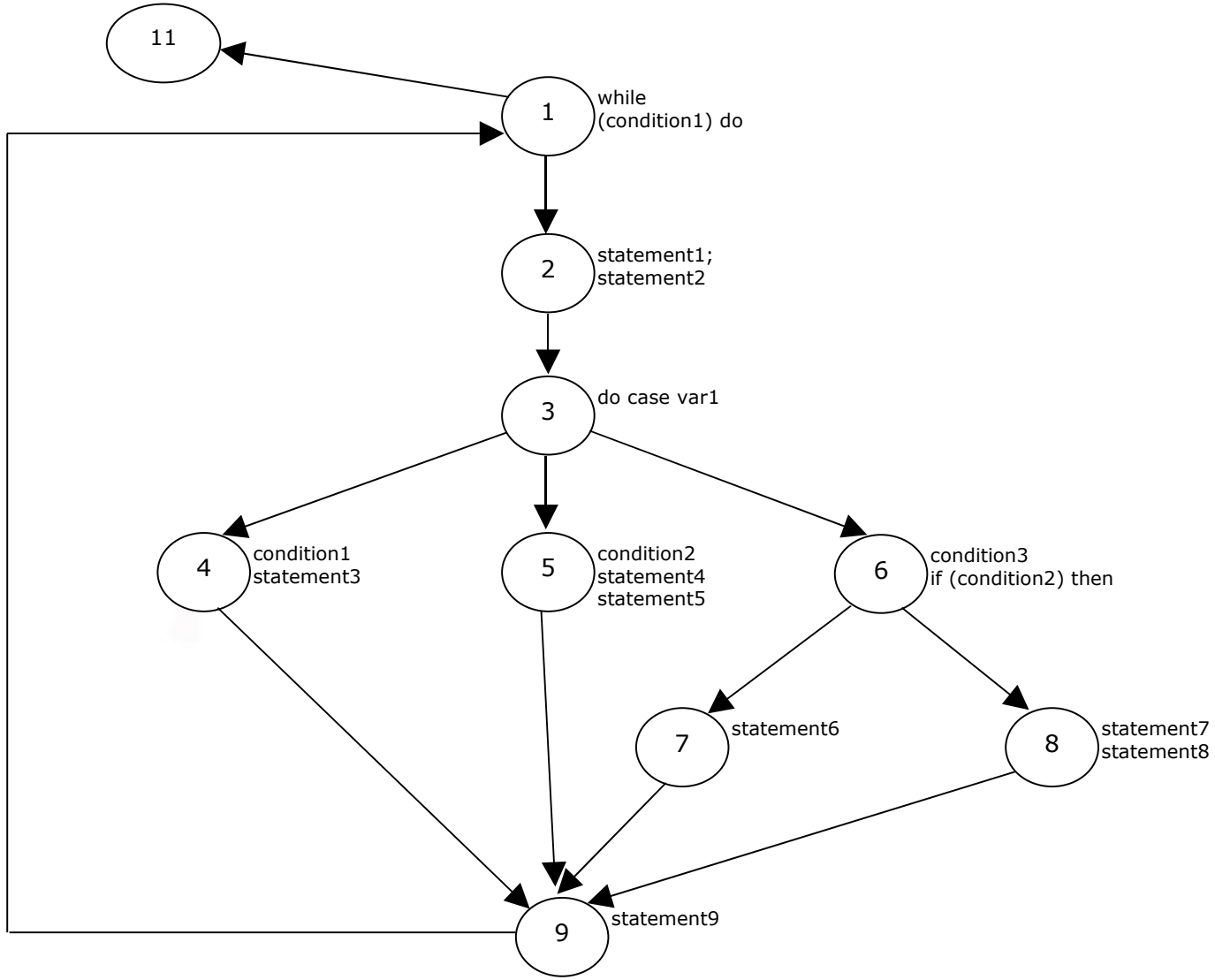
- Configuração primária é identificada para determinar os caminhos de execução

Etapa 1: Utilizar a especificação procedural

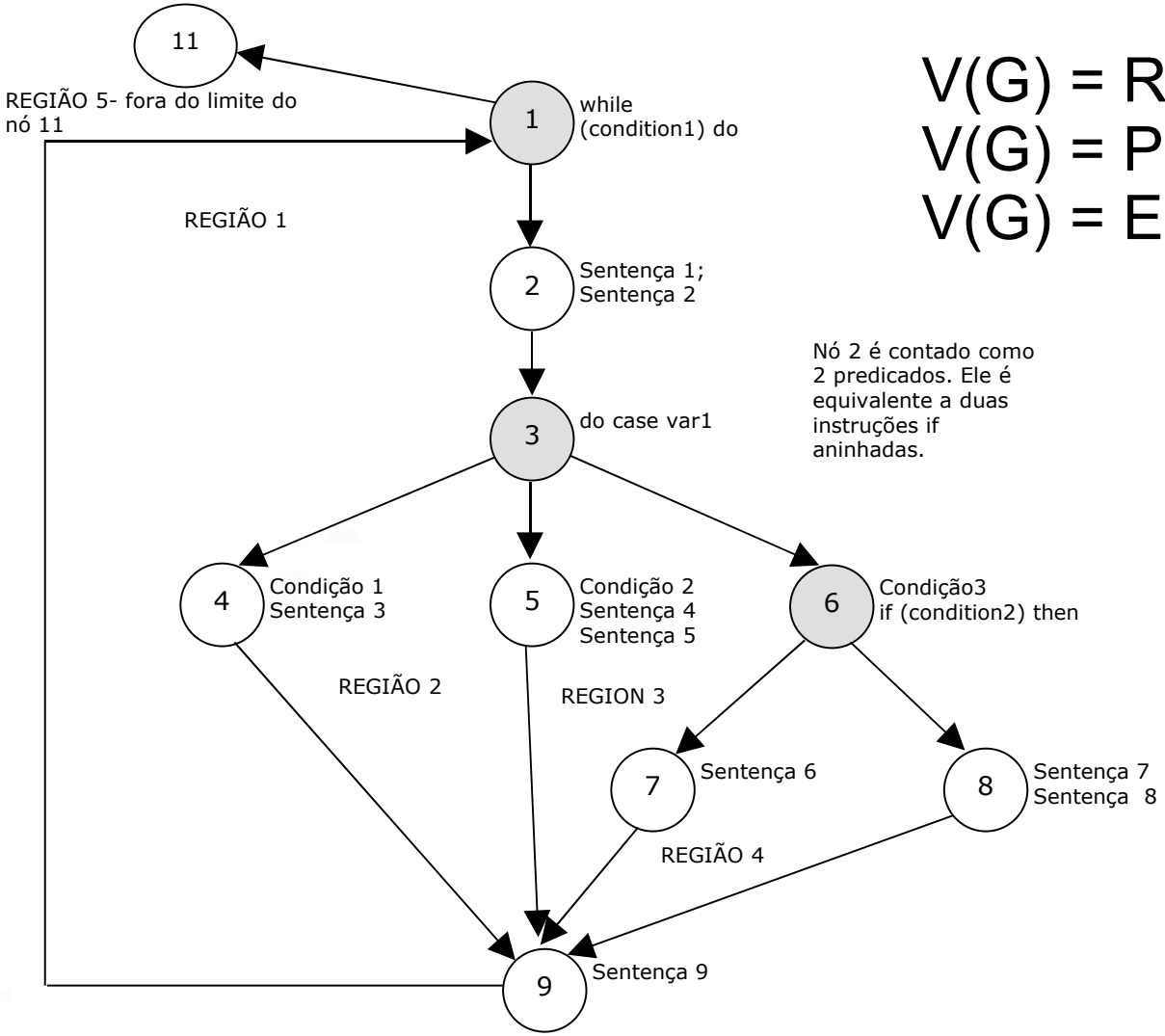
```
while (condition1) do
  statement1;
  statement2;
  do case var1
    condition1:
        statement3
    condition2:
        statement4;
        statement5;
    condition3:
        if (condition2) then
           statement6;
        else
             statement7
             statement8
        endif
  endcase
  statement9;
endwhile
```

Texto 1: Módelo de Código de Design

Etapa 2: Desenhar o Gráfico de Fluxo



Etapa 3: Calcular a Complexidade do Código



V(G) = R = 5 regiões
V(G) = P + 1 = 4+1 =5
V(G) = E-N+2 = 13-10+2 = 5

Etapa 4: Determinar os caminhos de execução

Caminho 1: Node-1, Node-2, Node-3, Node-4, Node-9

Caminho 2: Node-1, Node-2, Node-3, Node-5, Node-9

Caminho 3: Node-1, Node-2, Node-3, Node-6,Node-7, Node-9

Caminho 4: Node-1, Node-2, Node-3, Node-6, Node-8, Node-9

Caminho 5: Node-1, Node-11

Etapa 5: Documentar os Casos de Testes

Teste de Caso 1- Caminho 1:

para Node-1, a condição deve ser avaliada como sendo TRUE (identificar os valores necessários).

A avaliação de var1 deveria guiar para o Node-4 (identificar o valor de var1)

Resultado esperado: deverá produzir resultado suficiente para o Node-9

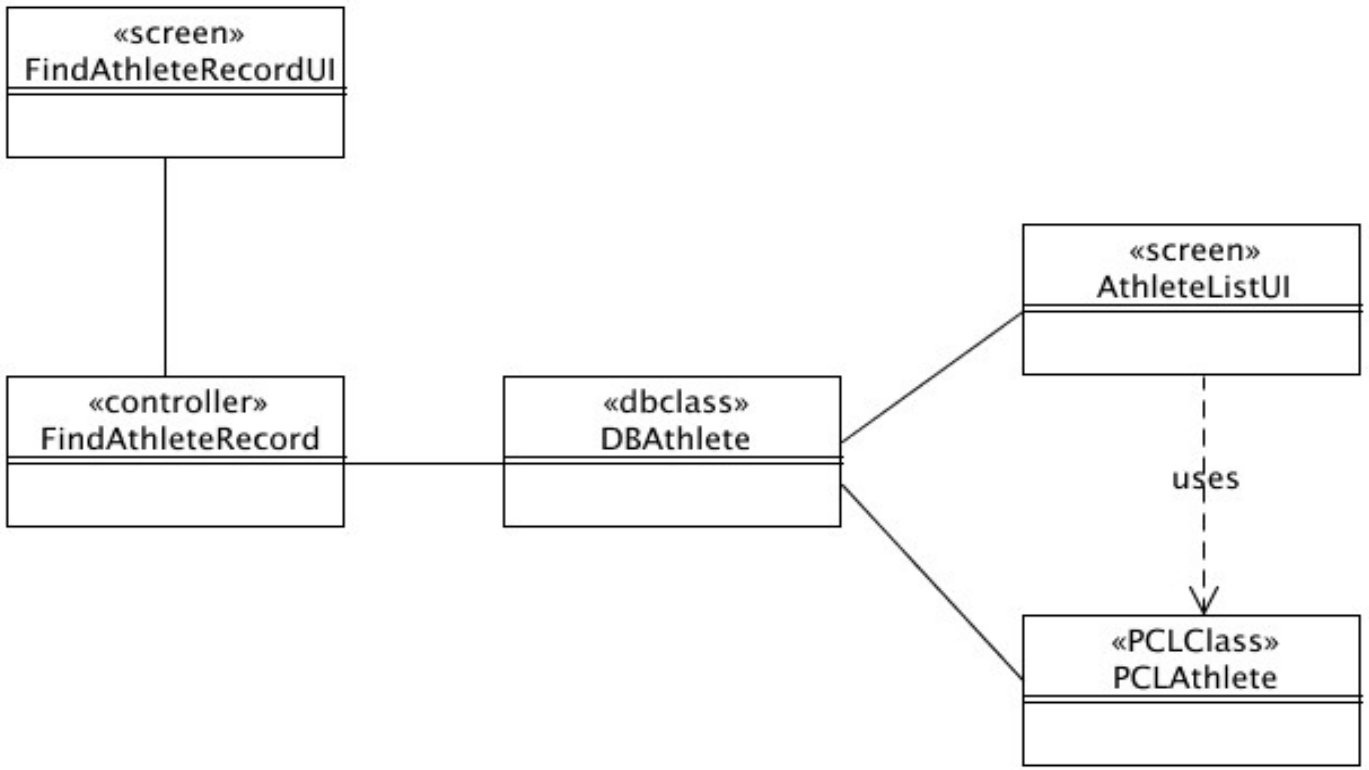
Teste das Estruturas de Controle

- Técnica de caixa-branca que testa os seguintes tipos de controle:
  - Testes de Condição
  - Testes de Repetição
  - Testes de Fluxo de Dados

Teste Caixa-preta

- Testar os aspectos funcionais do projeto
- Definir a série de testes de caso que identificam erros
- Técnicas:
  - Testes baseados em gráficos
  - Testes de Equivalência
  - Testes de limites de Valores

Testes Baseados em Gráficos: Passo 1



Testes Baseados em Gráficos: Passo 2

- Testes de Caso 1:
  - A classe FindAthleteUI envia uma solicitação para devolver uma lista de atletas baseada em um critério de busca. A solicitação é enviada para o FindAthleteRecord
  - O FindAthleteRecord envia uma mensagem ao DBAthlete para processar o critério de busca
  - O DBAthlete solicita ao servidor de banco de dados que execute a instrução SELECT. Isto preenche a classe PCLAthlete com a informação do atleta. Retorna a referência do PCLAthlete para o AthleteListUI
  - O AthleteListUI lista os nomes dos atletas

Testes de Equivalência

- Teste de equivalência é um teste caixa-preta que utiliza a entrada de dados do programa
- Isto divide a entrada de dados em grupos para que os testes de caso possam ser desenvolvidos
- Para isto se faz uso das classes de equivalência, que são regras que validam ou invalidam a entrada de dados

Teste de Limite de Valor

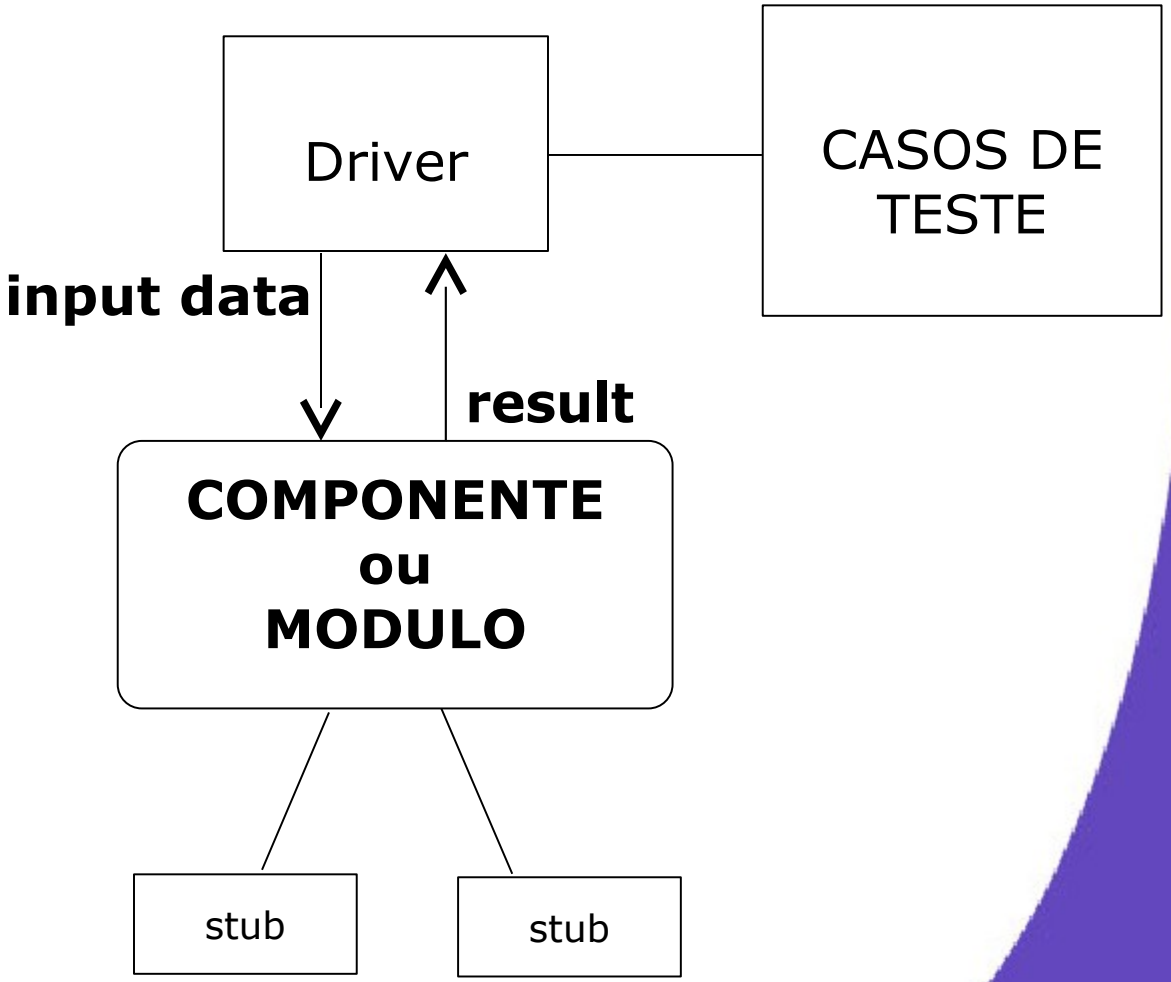
- Faz uso dos limites dos valor para se criar Casos de Teste
- Maioria das falhas ocorrem no limite dos valores de entrada

Testando Classes

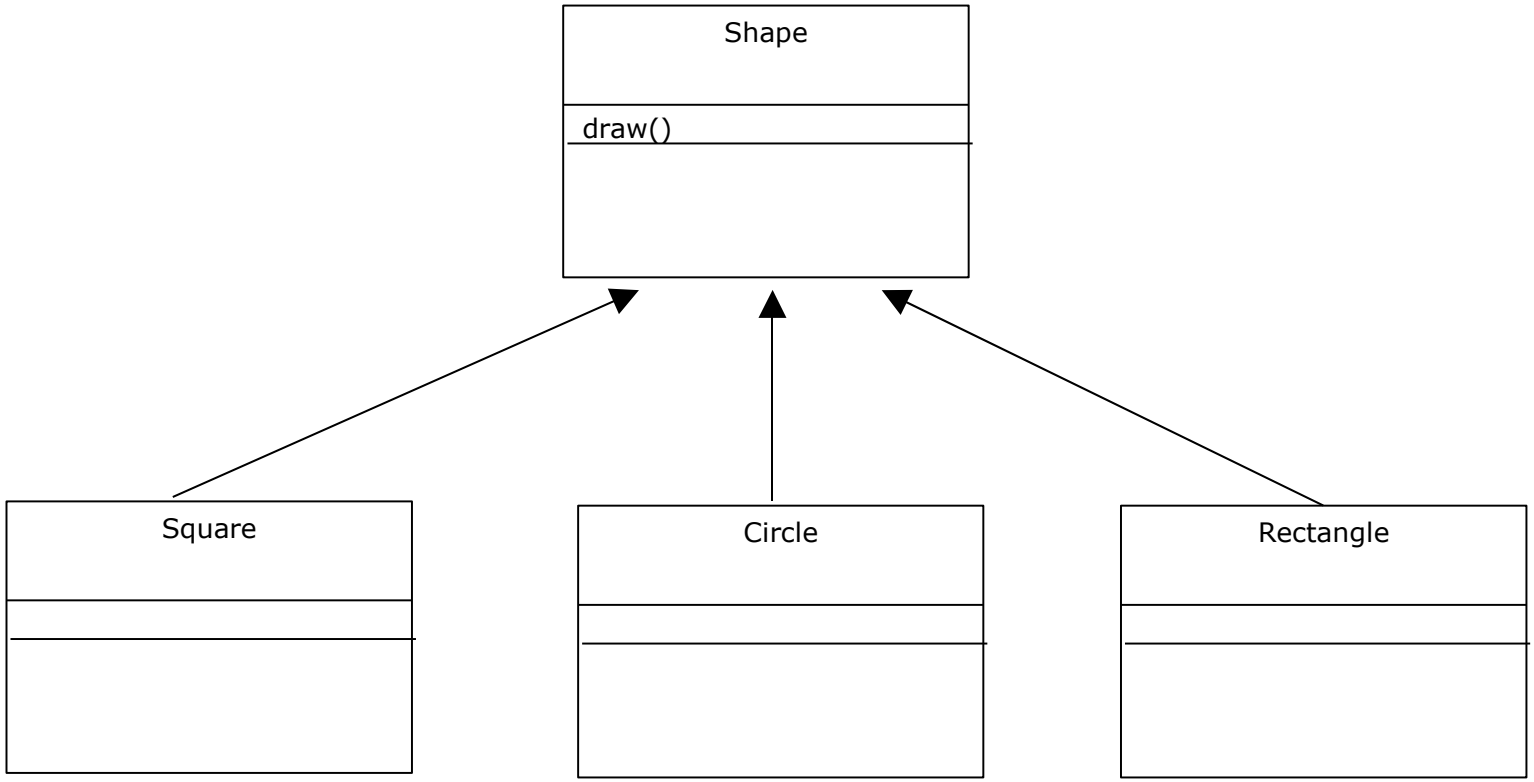
- Testes de Unidades
- Testes de Integração

Testes de Unidades

- Teste de unidade é o nível básico dos testes
- Tem como intenção se testar os menores blocos
- Processo no qual se executa cada módulo



Teste Orientado a Objetos



Teste de Integração

- Verificar se cada componente interage de maneira satisfatória
- A estrutura de controle da arquitetura do programa dita como os testes de integração deverão ser feitos

Teste de Aproximação baseado em Thread

- Grupo de classes que colaboram ou interagem quando uma entrada necessita ser processada ou um evento foi ativado
- Thread é um caminho de comunicação entre as classes
- Uso do Diagramas de Seqüência e Diagramas de Colaboração

Teste de Aproximação baseado em Uso

- Começam por identificar as classes independentes
- O próximo conjunto de classes a ser testado são chamadas de classes Dependentes
- *Clustering* pode ser usado
- *Clustering* é o processo que define um grupo de classes que podem ser integradas e testadas juntas
- Chamadas de *clusters* pois são consideradas como uma unidade

Metodologia de Desenvolvimento voltada para o Teste

- Adota a aproximação de primeiro testar somado a refazer
- Aproximação de se testar primeiro é uma técnica de programação, que envolve analise, projeto, codificação e teste
- Passos básicos envolvidos:
  - Escrever o necessário para se descrever o próximo incremento de comportamento
  - Escrever o código necessário para se passar no teste

Refatoração

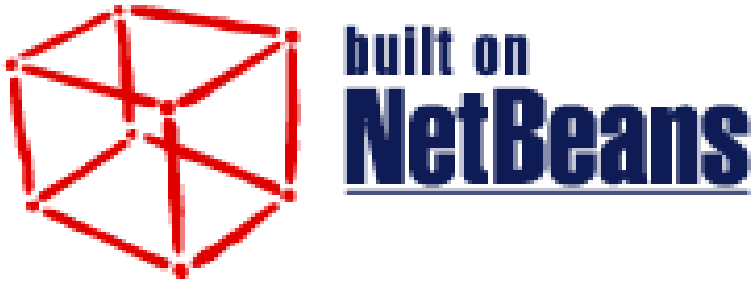
- Significa melhorar um projeto
- Eliminar partes redundantes ou duplicadas
- Processo no qual se modifica a estrutura interna do código sem que sejam feitas alterações no comportamento do programa

Beneficios do Desenvolvimento Voltado para Testes (TDD)

- Permitir o desenvolvimento simples de incremento
- Envolver um processo de desenvolvimento mais simples
- Prover um constante teste de regressão
- Melhorar a comunicação
- Melhorar no entendimento do comportamento do software
- Centralizar o conhecimento
- Melhorar o projeto

Passos para a TDD

- Passaremos agora para o NetBeans



Testes de Sistemas

- Depois do teste de integração, o sistema é testado como um todo com foco na funcionalidade e correção
- Deve incluir testes como o de performance, que testa o tempo de resposta e utilização de recursos
- **Teste de estresse**, que testa o software sujeito a quantidade, freqüência e volume de dados, pedidos e respostas anormais
- **Teste de segurança**, que testa os mecanismos de proteção do software
- **Teste de recuperação**, que testa o mecanismo de recuperação do software em caso de falha

Passos para a geração de Testes de Caso para Sistemas

- Utilizar o RTM e priorizar os casos de uso
- Para cada caso de uso, criar cenários
- Para cada cenário, obter pelo menos um caso de teste e identificar as condições de execução
- Para cada caso de teste, determinar os valores para os dados

Teste de Validação

- Começar depois do ápice do teste de sistema
- Consiste em uma série de casos de teste caixa-preta
- Critério de Validação

Testes Alfa & Beta

- *Testes Alfa & Beta* são uma série de testes de aceitação para permitir o usuário final validar todos os requisitos

- Um *Teste* Alfa é conduzido em um ambiente controlado, geralmente no ambiente de desenvolvimento

- *Testes Beta*, ao contrário, são conduzidos em um ou mais ambientes do usuário e os desenvolvedores não estão presentes

Componentes de Teste de Software RTM

Componentes de Teste de Software RTM	*Descrição*
Especificação de Testes	O nome do arquivo que contém o plano de como testar os componentes do software.
Casos de Teste	O nome do arquivo que contém os casos de teste para serem executados como parte da especificação de teste.

Métricas de Teste

- Falta de Coesão nos Métodos – *Lack of Cohesion in Methods (LCOM)*
- Porcentagem pública e protegida – *Percent Public and Protected (PAP)*
- Acesso Público aos Data Members – *Public Access To Data Members (PAD)*
- Número de Classes Raiz – *Number of Root Classes (NOR)*
- Número de Filhos - *Number of Children (NOC)* – e Profundidade de Árvore de Herança - *Depth of the Inheritance Tree (DIT)*

Sumário

- Testes de Software
- Princípios de Testes de Softwares
- Práticas nos testes de Caixa-branca e testes de Caixa-preta
- Testes de Unidades
- Testes de Integração
- Desenvolvimento Baseado em Testes (TDD)
- Passos na criação de Caso de Teste
- Testes e Critérios de Validação
- Testes Alfa & Beta
- Componentes de Teste de Software RTM
- Métricas de Teste

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**