- Teste em recipiente
    - recipientes são ativados para permitir o ambiente de teste
    - requer mecanismos para implementar e executar testes num recipiente

8ºSlide

Teste de Software

- Teste: Uma tentativa de provar que alguma caraterística especifica do software e/ou hardware está ausente de modo que este pode ser estabelecido por meios humanos perceptíveis.
- Teste de software: o processo de execução de um sistema de software para determinar se ele corresponde a sua especificação no ambiente pretendido.
- Especificações são cruciais: Estabelecer o que é o comportamento correto

Software bugs

Falhas são de código ou design de erros
Uma falha é uma discrepância com o comportamento observável pretendido (manifestação de falhas)

Testamos para:

- Obter informações valiosas sobre o processo de construção
    - Quão bem que estamos a fazer ?
    - Não é um processo de verificação para falhar !
- Gerir riscos (ganhar confiança nos resultados)
    - P.Ex: Temos cobertura de testes suficientes ?
- Responder à pergunta final
    - O produto está pronto para lançamento ?

A natureza dos testes

Teste ≠ Depuração

- A **depuração** é o processo que os developers passam por identificar a causa de erros ou defeitos no código e fazer correções.
- O **teste** é uma exploração sistemática de um componente ou sistema com o principal objectivo de encontrar e relatar defeitos. (não correção de defeitos)

Estática vs dinâmica

- **Static**: testes onde o código não é exercido
- **Dinâmico**: execução de software, utilizando os dados de entrada

Princípios gerais - a filosofia de testes

- Teste mostra a presença de erros
    - O teste não podem mostrar que o software é livre de erros
- Testes exaustivos é (geralmente) impossível
    - Teste todas as combinações possíveis de dados e caminhos de código em código?
    - Desafio: saber quando parar.
- Testes iniciais
    - Teste lhe dá informações valiosas sobre o processo de construção
    - Quanto mais cedo um problema (defeito) é encontrado, a menos que os custos para corrigir
    - Se o teste for atrasado, existe um risco real de que não será feito
- O teste é dependente do contexto
    - Teste diferente é necessário em circunstâncias diferentes

Níveis de teste

Teste Unidade

- O código escrito para a unidade atende à sua especificação, antes da sua integração com outras unidades
- O teste de unidade requer acesso ao código que está ser testado

Teste de Integralão

- Objetivo: expor os defeitos nas interfaces e nas interações entre componentes integrados no sistema
- Teste de objetos: Código de interface principalmente.

Teste de Sistema

- Concentra-se no comportamento do sistema inteiro / produto em um ambiente vivo representante.
- Objeto de teste: o sistema.

Teste de Funcionalidade

- Envolver os utilizadores finais para validar o sistema que irá funcionar de acordo com as suas expectativas
- Teste de objetos: o sistema totalmente integrado; formas e relatórios

É bom o suficiente? O teste de aceitação

Deve ser concluída com êxito

- ➔ Antes que o novo sistema possa ir ao vivo ou substituir um sistema legado.
- ➔ Conclusão pode ser um requisito contratual antes do sistema ser pago.

Caixa Fechado: pelo cliente

- ➔ Todo o sistema é testado como um todo
- ➔ A ênfase é sobre se o sistema atende aos requisitos
- ➔ Utiliza dados reais em situações reais, com utilziadores reais, administradores e operadores

Como testamos

O teste funcional

- ➔ Teste baseado em especificação aka, testes comportamentais, o teste de caixa-preta
- ➔ O teste funcional precisa de uma especificação

Testes não-funcionais

- ➔ Teste para as qualidades do sistema / atributos (por ex.: usabilidade, performance, ...)
- ➔ Precisa de uma especificação

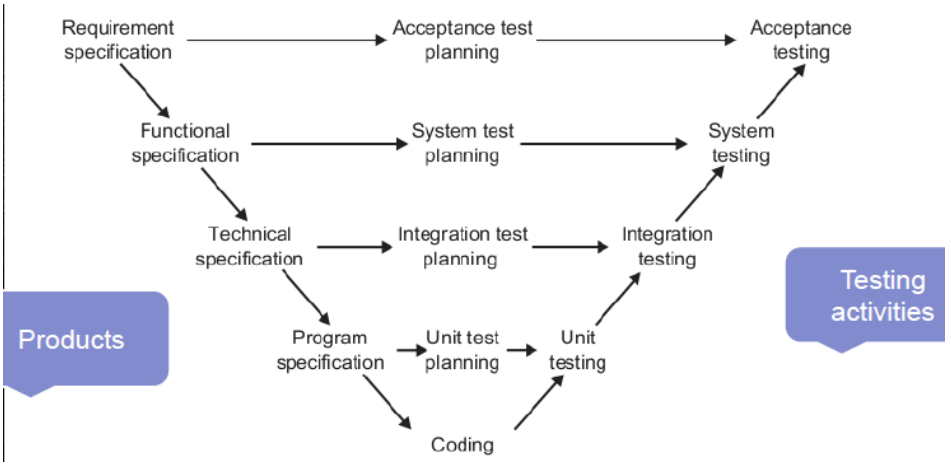
O teste estrutural

- ➔ Medir o quantos testes foram realizados contra algum conceito estrutural
- ➔ Por exemplo: user stories testadas/ todas as stories ; servlets / testados todos os servlets

Teste após o código alterado.

- ➔ Quando Hanges são intruduces, o sistema deve ser testado novamente
- ➔ Testes de regressão: verificações que corrige erros não apresentam funcionalidade inesperadas no sistema (correções foram bem sucedidos)

V Model



No V Model a execução dos processos acontece de forma sequencial em forma de V. Também é conhecido como um modelo de Verificação e Validação.

V - Model é uma extensão do modelo em cascata e baseia-se em associação de uma fase de testes para cada fase de desenvolvimento correspondente. Isto significa que, para cada fase do ciclo de desenvolvimento, há uma fase de teste diretamente associada. Este é um modelo altamente disciplinados e a próxima fase começa apenas após a finalização da fase anterior.

Vocabulário

Condição de Teste

Alguma caraterística do nosso software que pode verificar um teste ou um conjunto de testes (por exemplo, uma função, transação, caraterística, atributo de qualidade, ou elemento estrutural)

Caso de Teste

Recebe o sistema como ponto de partida para o ensaio (condição de execução); em seguida, aplica-se um conjunto de valores de entrada que deve alcançar um determinado resultado (resultado esperado), e sai do sistema como ponto final (pós condição de execução).

Procedimento de Ensaio (scripts de teste a.k.a. manuais)

As ações necessárias em sequência para executar um teste

Fluxo de trabalho de design de teste Essencial

1) Decidir sobre uma condição de teste, o que normalmente seria uma pequena parte da especificação para o nosso software em teste;
2) Projetar um caso de teste que vai verificar a condição de teste;
3) Escrever um procedimento de teste para executar o teste

Stories e Scenarios

Unidade básica da funcionalidade, e, portanto, de entrega.

- Captura uma caraterística do sistema
- Define o âmbito do recurso
- Os seus critérios de aceitação.

Título Como [papel] Eu quero [recurso] Assim que [beneficio]	Critérios de aceitação Given And When Then And

Código Fonte

- **Distribuído**: são vários repositórios autónomos e independentes para casa developer
- **Centralizado**: Segue a topologia de estrela, havendo apenas um único repositório central, mas existindo várias cópias de trabalho, uma para cada developer.

Cobertura

▪ Cobertura de teste é uma medida do grau ao qual um teste exerce alguma caraterística (s) ou de código.
- Minimizar o risco de não ter coberto todas as possibilidades de ter um bug fatal no sistema lançado
- Medidas podem fazer parte dos critérios de conclusão definidos no plano de teste e usados para determinar quando parar

Testes de Cobertura:
- A cobertura de teste de funcionalidade.
- A cobertura de teste de estrutura
- A cobertura de teste de cenário

Estratégias caixa preta vs branca

Caixa Preta	**Caixa Branca**
Várias estratégias - Divisão de equivalência - Análise do valor limite - Erro de adivinhação - Causa-efeito gráfico - Testes aleatórios - Testes de partição	Várias estratégias - Verificação formal - A revisão de código / inspeção - Cobertura ▪ testes Caminhos ▪ cobertura Decisões ramo ▪ testes de Dados de fluxo

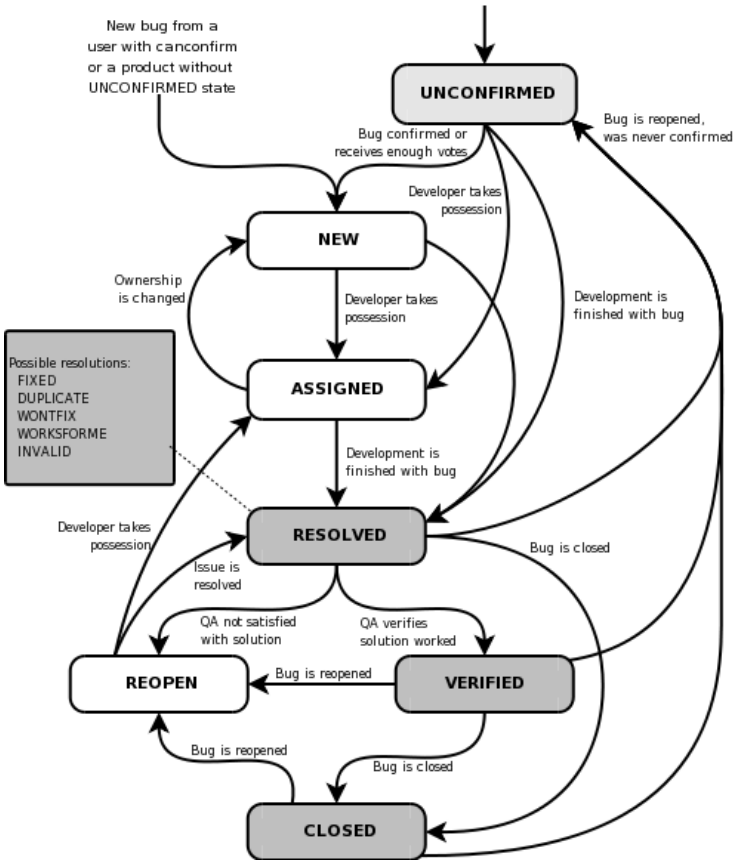
- Testes de valor de limite - Testes baseados em tabelas de decisão	▪ teste orientado a objeto

Exemplos de casos de teste seleções

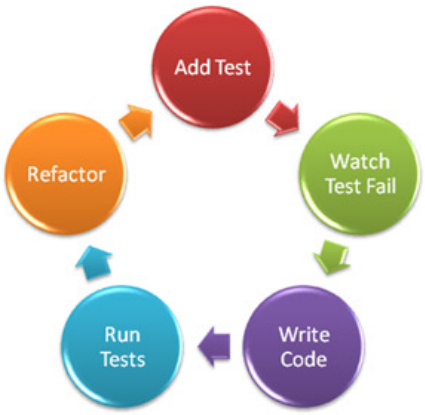
Cobertura de inputs - O objetivo é testar todas as classes de entrada - Aulas de dados - principais categorias de transacções de dados e entradas. ▪ Intervalos de dados - valores típicos, extremos ▪ Dados inválidos ▪ Reversões, recargas, reinicia após falha	Cobertura de todas as funções - O objetivo é testar todas as funções de cada programa de computador ▪ Caminhos através dos programas de computador - Análise de demonstração - Testes de ramo - Fluxo do Gráfico do Programa ▪ Verifique se todos os caminhos são executado pelo menos uma vez - Se todas as declarações e todos os ramos são testado é o programa correto?

Vida de um bug

- Erros são registrados / abertos
- Pode ser aceito / rejeitado
- Seja atribuído
- Código com correções apresentadas
- Retest
- Relatórios, gráficos de defeito, ...

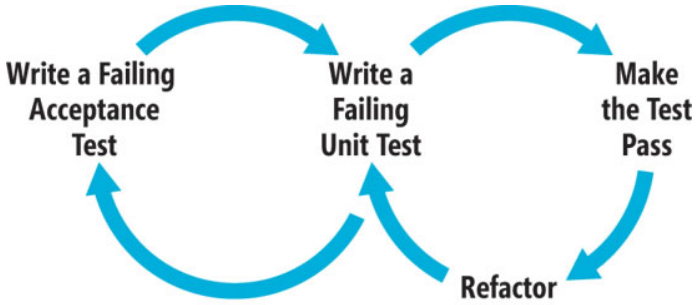


TDD (Test Drive Development): é uma técnica de desenvolvimento de software que se baseia num curto circulo de repetições.



1. Adiconar Testes
2. Verificar se ocorre falhas
3. Escrever código
4. Correr testes
5. Refatorar Código
6. Repetir

BDD (Behavior Drive Development): é uma técnica de desenvolvimento ágil que encoraja a colaboração de developers, testores de qualidade e pessoas não técnicas.



9°Slide

Botoom line

Inferno Integração Grande esforço, imprevisível para integrar estado da aplicação, não é executável na maioria das vezes.	**Integrar cedo e frequentemente** Propriedade do código de erros pontuais anteriormente partilhado todo mundo é co-responsável.

A essência do que está na prática simples onde toda a equipa se integra frequentemente.

Integração Contínua (CI)

- CI consiste em construir um projeto regular e automaticamente com execução de testes automatizados, testes de qualidade, mecanismos de integração e implementação de binários em máquinas de execução ou de partilhamento de repositórios.
- CI faz com que o processo de desenvolvimento seja mais suave, mais previsível e menos arriscado.

<u>Ferramentas:</u>