

UnB Gama
O novo endereço da Tecnologia.

Verificação e Validação de software

Prof. Ricardo Ajax
ricardoajax@unb.br

1

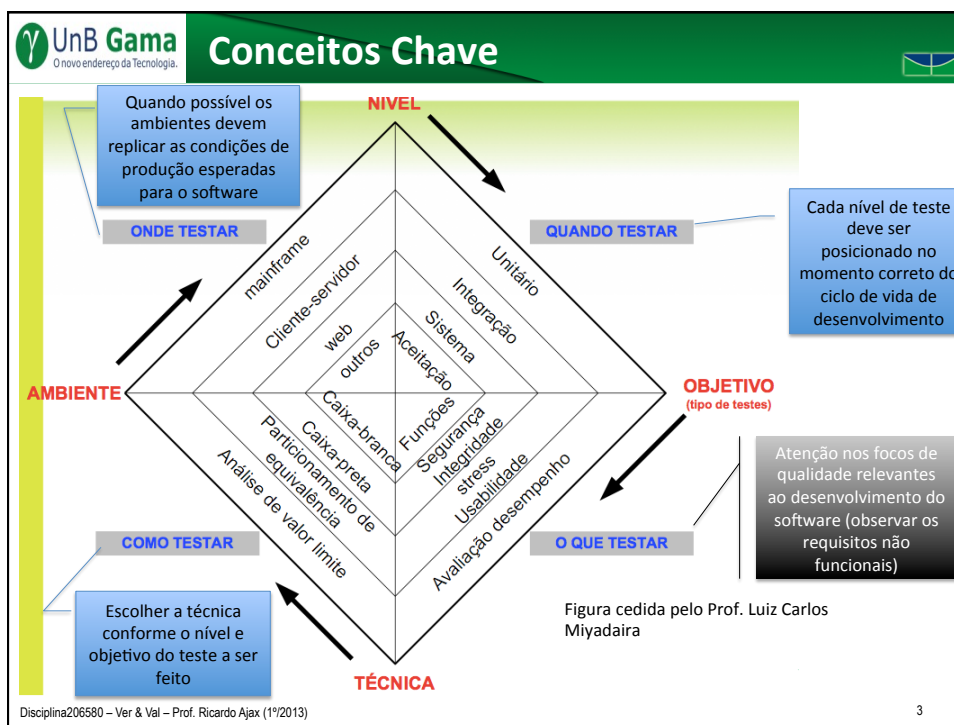
UnB Gama
O novo endereço da Tecnologia.

Conceitos Chave

Figura cedida pelo Prof. Luiz Carlos Miyadaira

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

2



UnB Gama O novo endereço da Tecnologia.


Níveis de Teste (ainda)

Então, para fechar este assunto:


- Benefícios da estruturação de testes em níveis:
 - Evita redundância de testes, cada nível trata uma classe específica de defeitos
 - Níveis de testes podem ser aplicados de acordo com o tamanho do software
 - Software para uso do próprio autor pode ir apenas até o nível de integração
 - Softwares comerciais utilizados em grande escala exigem testes em níveis mais elevados

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

4




Níveis de Teste




- Considerações
 - Os níveis não são necessariamente sequenciais
 - As características do sistema a ser testado irão determinar:
 - Níveis de teste
 - Tipos de testes
 - Técnicas de derivação de casos de teste

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 5



Fundamentos de teste



Níveis de testes

1. **Testes de unidade:**
 - têm como objetivo verificar o menor elemento testável de um software. Tipicamente aplicado para componentes representados no modelo de implementação (programas, componentes).
 - Geralmente realizado pelo programador. Testam o comportamento individual desses elementos
2. **Testes de integração:**
 - Objetivo: realizados para verificar se os componentes utilizados no modelo de implementação operam conforme especificado para realizar uma funcionalidade.
 - Testam o comportamento conjunto de elementos constituintes de uma determinada funcionalidade.
3. **Testes de sistema:**
 - Objetivos: tradicionalmente realizado quando o sistema funciona por completo e com base nas suas funcionalidades advindas dos requisitos da aplicação.
4. **Testes de aceitação:**
 - É o teste realizado antes da disponibilização do software ao seus usuários finais. Podem ser categorizados em:
 - Formais
 - Informais ou Alfa-Teste
 - Beta-Test

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 6

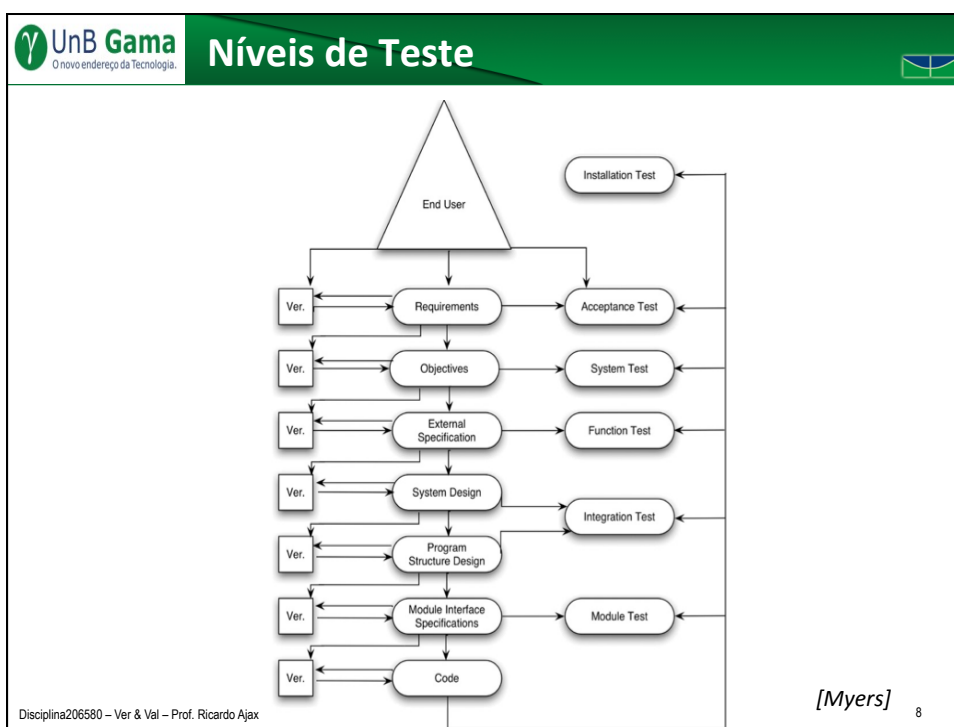
UnB Gama O novo endereço da Tecnologia.


Fundamentos de teste

Objetivos – O que testar (Tipos de testes)


Critério de Qualidade (FURPS+)	Tipos de Teste
Funcionalidade	Funções Segurança Volume
Usabilidade	Usabilidade
Confiabilidade	Integridade Estrutura Stress
Desempenho	Avaliação de Desempenho Carga
Suportabilidade	Configuração Instalação

7






Testes de Unidade




- É o processo de testar **individualmente** as menores partes testáveis de um determinado código.
- Têm como objetivo verificar **o menor elemento testável** de um software
 - Em linguagens procedurais uma unidade pode ser uma função ou um procedimento
 - Em linguagens orientadas a objeto uma unidade pode ser uma classe ou um método
- Geralmente realizado pelo **desenvolvedor**
- Casos de teste com foco nos algoritmos e lógica de programação
- Tipicamente **white-box**
- Propósitos:
 - Gerenciar os elementos combinados do teste, começando pela menor unidade
 - Facilitar o processo de depuração (localização do defeito)
 - Permitir paralelismo da atividade de teste

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

9



JUNIT



- É um **framework de teste unitário** de código aberto para Java, que permite escrever e executar testes automatizados
- Foi criada por Kent Beck e Erich Gamma
- É uma das APIs de teste unitário mais utilizadas
- Permite testar uma única classe ou criar uma *suite de testes* para testar um grupo de classes
- Integrada a uma ferramenta de geração de build (como o Apache Ant), permite executar **suítes de teste** como parte do processo de geração do build, gerando relatórios de resultados

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

10

UnB Gama
O novo endereço da Tecnologia.

JUNIT

- Uma **suite de testes** (*TestSuite*) contém uma coleção de testes que podem ser mais **suites de teste** (*TestSuites*) ou **casos de teste** (*TestCases*), ou classes implementando a interface de testes

```

classDiagram
    class Test {
        <<interface>>
        run(ResultSet)
    }
    class TestCase {
        setup()
        tearDown()
    }
    class TestSuite {
    }
    Test <|-- TestCase
    TestSuite o--> TestCase
          
```




Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
11

UnB Gama
O novo endereço da Tecnologia.

JUNIT

- Uma **suite de testes** (*TestSuite*) contém uma coleção de testes que podem ser mais **suites de teste** (*TestSuites*) ou **casos de teste** (*TestCases*), ou classes implementando a interface de testes
- Um teste completo é realizado em duas fases:
 - 1) Configuração:
 - Criação de uma hierarquia de objetos de casos de teste
 - Cada objeto representa um caso de teste e é uma instância de uma subclasse da classe *TestCase*
 - Na implementação das subclasses, os métodos iniciados por “test” contém o código de teste
 - 2) Execução dos casos de teste:
 - A execução é iniciada quando o método *run* do objeto raiz da hierarquia é chamado
 - A travessia na árvore é realizada por profundidade, sendo que a ordem dos nós é a de criação
 - O resultado da execução é armazenado em um objeto da classe *TestResult*




Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
12

- No desenvolvimento dos casos de teste as asserções são utilizadas para verificar se o resultado correto está sendo retornado
- Um asserção é uma expressão booleana em um ponto específico de um programa que retorna Verdadeiro a menos que exista um bug no programa.
- Todas as asserções estão na classe assert que provê um conjunto de métodos uteis para a escrita de testes. Somente são registradas asserções que falham.
- Vários tipos de assert podem ser utilizados. Alguns exemplos são:
 - `assertEquals(expected, actual)` → Verificam se duas primitivas (objetos) são iguais;
 - `assertTrue(boolean condition)` → Verifica se uma determinada condição é verdadeira
 - `assertFalse(boolean condition)` → Idem (Falsa)
 - `assertNull(objeto) / assertNotNull(Objeto)` → Verifica se um determinado objeto é nulo/não nula
 - Existem várias outras
- Após a execução, o Junit irá apresentar:
 - Vermelho, se o teste falhar / Verde, se o teste passar

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

13


- Cuidado: Asserções são também códigos. E podem ter erros. O que acontece se elas tiverem defeitos e falharem?
 - Reportarem um erro quando ele não existe realmente
 - (menos mal)
 - Falhar no reporte de um bug que efetivamente existe
 - Uma boa ideia é simular situações de erros para avaliar se as asserções estão OK
- Além disso, podem causar efeitos colaterais.
 - Ocupam tempo para serem executadas
 - Consomem memória para serem executadas

Dependem da complexidade das asserções.


Mas constituem-se em uma boa estratégia para executar testes.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

14

**UnB Gama**
O novo endereço da Tecnologia.


JUNIT




- Ao criar uma classe de teste para uma classe qualquer, a (IDE) – Integrated Development Environment, são criados automaticamente:
 - Um caso de teste para cada método da classe sendo testada
 - Contendo a anotação `@Test`, que indica que o método deve ser executado pelo framework de teste
 - Quatro métodos padrão:
 - `@BeforeClass`
`setUpClass()`
 - `@AfterClass`
`tearDownClass()`
 - `@Before`
`setUp()`
 - `@After`
`tearDown()`

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

15

**UnB Gama**
O novo endereço da Tecnologia.


JUNIT




- `@BeforeClass`
`setUpClass()`
 - Pode incluir código que deve ser executado antes do teste
 - Exemplos:
 - Criação de conexão com banco de dados
 - Leitura de arquivo
 - A anotação `@BeforeClass` pode ser usada em outros métodos para os quais se queira que executem antes dos casos de teste

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

16

**UnB Gama**
O novo endereço da Tecnologia.


JUNIT




- **@AfterClass**
tearDownClass()
 - Inclui o código que deve ser executado após todos os casos de teste
 - Utilizado para liberar os recursos adquiridos no método *setUpClass*
 - Exemplos:
 - Fechamento da conexão com o banco de dados
 - Liberação de arquivos
 - A anotação **@AfterClass** pode ser usada em outros métodos para os quais se queira que executem após todos os casos de teste

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

17

**UnB Gama**
O novo endereço da Tecnologia.


JUNIT




- **@Before**
setUp()
 - Inclui o código que deve ser executado antes de cada caso de teste
 - Utilizado para inicializar recursos antes de cada caso de teste
 - A anotação **@Before** pode ser usada em outros métodos para os quais se queira que executem antes de cada casos de teste
 - **@After**
tearDown()

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

18

**UnB Gama**
O novo endereço da Tecnologia.


JUNIT




- **@After**
tearDown()
 - Inclui o código que deve ser executado após cada caso de teste
 - Utilizado para liberar recursos após cada caso de teste
 - A anotação **@After** pode ser usada em outros métodos para os quais se queira que executem após cada casos de teste

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

19

**UnB Gama**
O novo endereço da Tecnologia.

Testes de Integração



1. O teste de integração é executado para garantir que os **componentes de software** funcionem corretamente quando **combinados**.
2. O objetivo do teste de integração é detectar imperfeições ou erros nas especificações das **interfaces** dos componentes integrados.
3. A estratégia de integração Pode ser **top-down**, ou **bottom-up**, ou **mista** e deve ser baseada no grau de acoplamento e coesão dos componentes sendo integrados


Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

20

UnB Gama
O novo endereço da Tecnologia.

Testes de Integração

1. Duas situações:




Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

21

UnB Gama
O novo endereço da Tecnologia.


Testes de Integração

1. Duas situações:




Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

22

**UnB Gama**
O novo endereço da Tecnologia.


Testes de Integração




- De acordo com a **estratégia de integração**, testes são necessários sem que se tenham disponíveis todos os componentes de software que compõem o produto integrado.
- Nestes casos são criados Componentes de Teste de Integração (**Drivers e Stubs**) que **simulam** o funcionamento e a interface do produto de software sendo testado.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

23

**UnB Gama**
O novo endereço da Tecnologia.



Testes de Integração



- Na estratégia **top-down** inicia-se a integração a partir dos níveis mais altos da hierarquia de controle
 - Necessária a utilização de **stubs** para substituir os componentes ainda não integrados



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

24

**Testes de Integração**


- Stub ou Method stub (esboço de método)
 - Geralmente, os componentes de software dependem de outros componentes para concluir suas tarefas
 - Os problemas surgem quando esses componentes secundários não são operacionais. Às vezes, ainda estão em desenvolvimento, ou então têm muitos erros.
 - O teste dos componentes principais não precisa ser interrompido até que os componentes secundários estejam disponíveis

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 25


**Testes de Integração**

- Stub ou Method stub (esboço de método)
 - Em vez disso, um **stub** ou componente temporário pode substituir qualquer componente não operacional para fins de teste.
 - O stub não implementa a funcionalidade do componente real, ele simplesmente reage a entradas.
 - Os stubs retornam uma resposta programada para um determinado conjunto de valores, sem implementar qualquer lógica. É um simples relacionamento de estímulo/resposta

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 26

**UnB Gama**
O novo endereço da Tecnologia.

Testes de Integração



- Na estratégia **bottom-up** inicia-se a integração pelos componentes de níveis mais baixos, seguindo em direção aos de níveis mais altos da hierarquia de controle
 - Necessária à utilização de **drivers** para substituir os componentes ainda não integrados.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

27

**UnB Gama**
O novo endereço da Tecnologia.

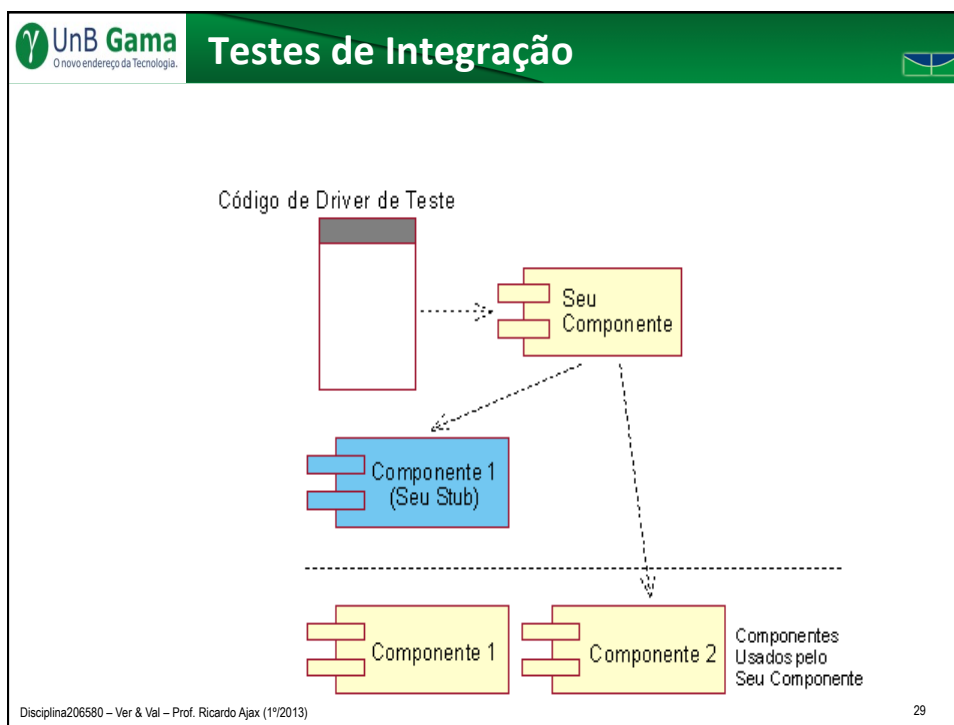
Testes de Integração



- Drivers são componentes de software usados para
 - disparar um teste e, muitas vezes, fornecer dados de teste, controlar e monitorar execução e relatar resultados de teste.
 - O driver de teste controla a execução de um ou mais testes, passando informações para o produto ou componente de software sendo alvo do teste.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

28



UnB Gama
O novo endereço da Tecnologia.

Testes de Integração

- As seguintes situações motivam a criação de Drivers e Stubs:
 - O componente encontra-se **em desenvolvimento** ou ainda não foi implementado;
 - O componente possui **defeitos** que impeçam o funcionamento dos testes ou que fazem o testador perder muito tempo descobrindo que uma falha de teste não foi causada pelo componente;
 - O componente pode **dificultar a execução** dos testes. Caso o componente insira restrições ao teste como janela de tempo de execução, autenticação de usuário, etc.



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

30

**Fundamentos de teste**


- **Tipo:** Testes Funcionais
- **Propósito:** Testes destinados a validar as funções do produto ou componente de software conforme as especificações de requisitos funcionais.
- **Ocorrência:** Unidade, Integração, Sistema e Aceitação


Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 31

**Testes Funcionais**

- Em alguns casos denominados **Teste de Sistema**
- Realizado para verificar se os componentes utilizados no modelo de implementação **operam conforme especificado** para realizar uma funcionalidade
- Propósito de encontrar **discrepâncias** entre o software e a especificação mais detalhada dos requisitos
- Normalmente realizado pela **equipe de desenvolvimento**
- Normalmente **black-box**

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 32


**UnB Gama**
O novo endereço da Tecnologia.


Testes Funcionais

- **Casos de teste** são derivados da especificação detalhada dos requisitos
- **Exemplos de métodos de derivação:**
 - Particionamento de equivalência
 - Análise de valor limite
 - Grafo de causa-efeito

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

33



**UnB Gama**
O novo endereço da Tecnologia.

Testes Funcionais

- **Recomendações:**
 - Identificar funcionalidades que geram maior quantidade de erros → tendência a ter mais erros não identificados
 - Dar a devida atenção às entradas de dados inválidas ou inesperadas
 - A definição do resultado esperado do caso de teste é fundamental



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

34

**Testes de Sistema**


- O **software é apenas um elemento de um sistema** baseado em computador mais amplo
- O *teste de sistema* envolve uma série de diferentes testes, cujo propósito primordial é por completamente à prova o sistema baseado em computador
- Tradicionalmente realizado quando o sistema funciona por completo
- Propósito de comparar o sistema com seu objetivo original
 - Necessidade de se estabelecer os objetivos, de preferência mensuráveis, do sistema
 - Casos de testes projetados de acordo com os objetivos e não de acordo com as especificações [Myers]
- Geralmente realizado por uma equipe independente

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 35


**Testes de Sistema**

- Não há método específico para derivação de casos de teste neste nível
- Em geral são aplicadas categorias ou tipos de teste como:
 - Teste de volume
 - Teste de stress
 - Teste de performance
 - Teste de instalação
 - ...

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 36

**UnB Gama**
O novo endereço da Tecnologia.


Teste de Aceitação




- É o teste realizado antes da disponibilização do software
- Normalmente realizado pelo cliente ou usuários finais
- No caso de contratação de software, o contratante compara o sistema com os requisitos do contrato
- Casos de teste voltados para os requisitos do contrato

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

37

**UnB Gama**
O novo endereço da Tecnologia.



Fundamentos de teste



- **Tipo:** Testes de Segurança
- **Propósito:** Testes destinados a garantir que o produto ou componente de software possa ser acessado apenas por determinados perfis de usuários ou atores. Esse teste é implementado e executado principalmente nos componentes de segurança do software, como os que realizam *login* do usuário
- **Ocorrência:** Unitário: no teste específico do componente de segurança; Sistema: avaliando as regras gerais de acesso e Aceitação.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



38

**Fundamentos de teste**

- **Tipo:** Testes de Volume
- **Propósito:** Teste destinado a verificar a capacidade do produto ou componente de software em lidar com um grande volume de dados, como entrada e saída ou residente no banco de dados. O teste de volume abrange estratégias de teste, como, por exemplo, a entrada de dados do volume máximo de dados em cada campo ou a criação de consultas que retornem todo o conteúdo do banco de dados ou que tenham tantas restrições que nenhum dado seja retornado.
- **Ocorrência:** Unitário: no teste específico do componente de acesso aos dados; Sistema: avaliando os requisitos gerais de volume. Aceitação

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



39

**Fundamentos de teste**

- **Tipo:** Testes de Usabilidade
- **Propósito:** Testes que enfatizam: fatores humanos, estética, consistência na interface do usuário, ajuda on-line e contextual, assistentes e agentes, documentação do usuário e material de treinamento.
- **Ocorrência:** Unitário: enquanto prova de conceito para avaliação dos requisitos de interface do software; Sistema: através da avaliação geral dos padrões de interface especificados e Aceitação.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



40

**Fundamentos de teste**

- **Tipo:** Testes de Integridade
- **Propósito:** Testes destinados a avaliar a robustez do produto ou componente de software (resistência a falhas) e a compatibilidade técnica em relação à linguagem, sintaxe e utilização de recursos.
- **Ocorrência:** Unitário: enquanto prova de conceito arquitetural e
- Integração.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



41

**Fundamentos de teste**

- **Tipo:** Testes de Estrutura
- **Propósito:** Testes destinados a avaliar a adequação do produto ou componente de software em relação a seu design e sua formação. Em geral, esse teste é realizado em aplicativos habilitados para a Web, garantindo que todos os links estejam conectados, que o conteúdo apropriado seja exibido e que não haja conteúdo órfão.
- **Ocorrência:** Sistema e Aceitação

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



42

**Fundamentos de teste**

- **Tipo:** Testes de Stress
- **Propósito:** Tipo de teste de confiabilidade destinado a avaliar como o sistema responde em condições anormais.
- O stress no sistema pode abranger cargas de trabalho extremas, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados.
- Deve ser planejado para que a carga seja constantemente aumentada até que o desempenho do sistema se torne inaceitável.
- Normalmente, esses testes são executados para compreender melhor como e em quais áreas o sistema será dividido, para que os planos de contingência e a manutenção de atualização possam ser planejados e orçados com bastante antecedência.
- **Ocorrência:** Sistema

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



43

**Fundamentos de teste**

- **Tipo:** Teste de Avaliação de Desempenho
- **Propósito:** Tipo de teste de desempenho que compara o desempenho do produto ou componente de software (novo ou desconhecido) a um sistema e uma carga de trabalho de referência conhecidos.
- Requer instrumentação do sistema para monitorar o uso dos recursos, tempos de resposta para determinar as situações que levam à degradação do desempenho.
- **Ocorrência:** Unitário: quando prova de conceito para avaliação de desempenho e Sistema: abrangendo o desempenho geral do software.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



44

**Fundamentos de teste**

- **Tipo:** Teste de Carga
- **Propósito:** Tipo de teste de desempenho usado para validar e avaliar a aceitabilidade dos limites operacionais de um sistema de acordo com cargas de trabalho variáveis, enquanto o sistema em teste permanece constante. Em algumas variáveis, a carga de trabalho permanece constante e a configuração do sistema em teste é que varia.
- **Ocorrência:** Unitário: quando prova de conceito para avaliação de desempenho; Sistema

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)


45

**Fundamentos de teste**


- **Tipo:** Teste de Configuração
- **Propósito:** Teste destinado a garantir que o produto ou componente de software funcione conforme o esperado em diferentes configurações de hardware e/ou software.
- Normalmente utilizado quando requisitos não funcionais de portabilidade fazem parte do sistema, como a necessidade visualizar a interface Web em diferentes fabricantes e versões de navegadores.
- **Ocorrência:** Unitário: quando prova de conceito para avaliação de desempenho; Sistema

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

46



Fundamentos de teste




- **Tipo:** Teste de Instalação


- **Propósito:** Teste destinado a garantir que o produto ou componente de software do teste seja instalado conforme o esperado em diferentes configurações de hardware e/ou software e sob diferentes condições (como no caso de espaço insuficiente em disco ou interrupção de energia). Esse teste é implementado e executado em aplicativos e sistemas.

- **Ocorrência:** Sistema

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
47



Fundamentos de teste

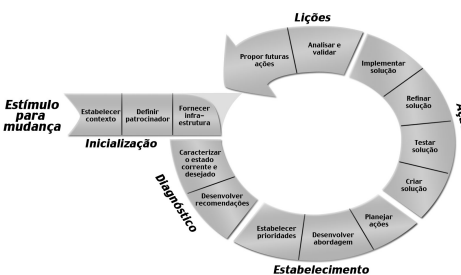


- **Não faltou nada não?**

- **E quando se quer melhorar um software?**

- **O que é melhorar um software?**

- **Que tipos de melhorias são costumeiramente solicitadas**



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
48

UnB Gama
O novo endereço da Tecnologia.

Fundamentos de teste

- Não faltou nada não?
- E quando se quer melhorar um software?
- O que é melhorar um software?
- Que tipos de melhorias são costumeiramente solicitadas

Manutenções

- Corretivas
- Adaptativas
- Perfectivas

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

49

UnB Gama
O novo endereço da Tecnologia.



Fundamentos de teste

Manutenções

- **Corretivas**
Manutenção para corrigir falhas (Bugs) no hardware ou software.
A modificação reativa de um software realizada depois da entrega para corrigir problemas descobertos. A modificação corrige o software para satisfazer requisitos. (ISO/IEC 14764:2006)
- **Adaptativas**
Manutenção para fazer com que o software continue sendo utilizável em um ambiente alterado.
Manutenção adaptativa fornece as melhorias necessárias para adaptar as modificações no ambiente em que o software deve funcionar. Essas mudanças são aquelas que devem ser realizadas para regular com o ambiente em alteração. Por exemplo, o sistema operacional deve ser atualizado e algumas alterações podem ser feitas para adaptar o novo sistema operacional. (ISO/IEC 14764:2006)

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

50


Fundamentos de teste


Manutenções



- Perfectivas**

Manutenção para melhorar a performance, facilidade de manutenção ou outros atributos do software instalado.

Manutenção de um software após a entrega para detectar e corrigir falhas ocultas no software antes que elas se manifestem como falhas. Manutenção perfectiva fornece melhorias para o usuário, melhoria da documentação do programa, e recodificação para melhorar a performance, manutenção ou outros atributos do software (ISO/IEC 14764:2006).

Manutenções evolutivas: Um subconjunto das perfectivas. Constituem-se em mudanças solicitadas pelos usuários de negócios que podem incluir, alterar ou mesmo excluir funcionalidades de um determinado software.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
51


Fundamentos de teste


- E quando é solicitada uma mudanças em alguma coisa?**
- GC (nível 2) x RD (nível 3) x REQM (Nível 2)**

Existem requisitos da aplicação

Existe uma gerência dos requisitos



Mesmo em níveis de maturidade mediana, a rastreabilidade dos requisitos deve percorrer desde os fornecedores de requisitos (usuários) até o código

Se existe uma mudança, uma avaliação do seu impacto deve(ria) ser feita para verificar o que deve ser modificado (GC – Gestão de configuração e mudança)

As modificações impactam em quais outras partes do software

Teoricamente: tudo o que for mudado deve ser (re) testado + tudo o que foi impactado pela mudança também deve(ria) ser (re)testado.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)
52

**Fundamentos de teste**

Em todos os casos de manutenções

Em todos os casos de mudanças



Quais são os impactos no restante do software?

Como saber se tudo continuará funcionando?

Testes de regressão

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



53

**Testes de regressão**

- Testes de regressão (*regression Testing*), devem ser executados depois de uma melhoria ou reparo em programas do sistema.
- Seu propósito é verificar se a mudança realizada em um componente ou programa causou impacto em outros funcionamentos do sistema, pois alterações podem “injetar” novos erros no software.
- São feitos por meio da (re)execução de um (sub)conjunto de testes já realizados
- Devem ser plenamente planejados com relação às suas execuções (o que, quem, como, quando, porque)
- Aqui a automação se faz bastante útil. Quais os critérios para realizar automações de testes funcionais? (uma boa questão de pesquisa).
- A grande pergunta é: Até onde regredir?
- Boa questão de pesquisa: E como isso tem sido feito em ambiente ágil de desenvolvimento de software. O que tem sido feito? Como tem sido controlado?

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)



54

**Fundamentos de teste**

Técnica de testes – Como testar

- Testes de Caixa-Branca
 - Exercitam a estrutura interna de um componente
 - Execução de cada bloco no mínimo uma vez
 - Execução de cada ramo condicional
 - Execução de caminhos com valores dentro dos limites
 - Execução de caminhos com valores fora dos limites
 - Verificação da estrutura de dados
 - Verificam se o projeto de software é válido e que o código foi implementado como projetado



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 55

**Fundamentos de teste**

Técnica de testes – Como testar

- Testes de Caixa-Preta
 - É conduzido com componentes integrados. Valida se o software atende aos requisitos externos sem levar em conta os caminhos de execução tomados para atender cada requisito.
 - O processo de teste de caixa-preta inclui a validação:
 - da integridade funcional em relação a estímulos externos
 - de todas as interfaces externas (incluindo humanas) através de uma faixa de condições normais e anômalas
 - da capacidade do sistema se recuperar ou minimizar o efeito de condições inesperadas
 - da habilidade do sistema em condições de stress

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 56

**Fundamentos de teste**



Técnica de testes – Como testar

Particionamento de equivalência

- É uma técnica que pode ser aplicada em qualquer nível de teste
- É uma boa técnica para ser aplicada primeiro ao se elaborar casos de testes
- Muitas vezes é aplicada sem a consciência de que está sendo utilizada
- O seu uso formal pode dar melhores resultados e economizar o esforço gasto em testes.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

57

**Fundamentos de teste**



Técnica de testes – Como testar

Particionamento de equivalência

- A idéia é subdividir as entradas em grupos que tenham comportamentos semelhantes e podendo ser tratados da mesma forma
- Se for necessário deve-se testar mais de um valor do intervalo considerado, principalmente se se quer testar valores típicos fornecidos pelos usuários esperados da aplicação
- É necessário testar apenas uma condição de cada partição, pois assume-se que todas as condições de uma partição serão tratadas da mesma forma pelo Software
- Partições válidas e inválidas devem ser consideradas

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

58

**Fundamentos de teste**



Técnica de testes – Como testar

Particionamento de equivalência

Passo a passo de uma boa identificação de partição de equivalência:

- Decompor o programa em funções (ou conjunto de programas no caso de funcionalidades integradas e implementadas por vários programas / componentes)
- Identificar variáveis e comportamentos de cada função
- Particionar os valores de cada variável em classes de equivalência (válidas e inválidas)
- Especificar os casos a serem testados, excluindo os casos impossíveis ou desinteressantes.

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 59

**Fundamentos de teste**

Técnica de testes – Como testar



Particionamento de equivalência - Exemplo

Um programa valida um campo numérico da seguinte maneira:

- Valores inferiores ou iguais a 1 são rejeitados
- Valores maiores que 1 e menores a 130 são aceitos
- Valores maiores que 130 são também rejeitados

• Qual a melhor classe de equivalência deve ser estabelecida para os testes desse programa?

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013) 60

**Fundamentos de teste**

Técnica de testes – Como testar

Particionamento de equivalência - Exemplo

Um programa valida um campo numérico da seguinte maneira:



- Valores inferiores ou iguais a 1 são rejeitados
- Valores maiores que 1 e menores a 130 são aceitos
- Valores maiores que 130 são também rejeitados

• Qual a melhor classe de equivalência deve ser estabelecida para os testes desse programa?

Valores entre 0,1 a 131. Porque satisfazem as condições válidas e inválidas:
 ≤ 1 ; entre 1 e 130; e > 130

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

61

**Fundamentos de teste**

Técnica de testes – Como testar

Análise do valor limite



Limites são áreas mais propensas a erros

Limites em partições de equivalência são seus valores mínimos e máximos.

Lembrar que devem ser testados valores mínimos e máximos

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

62

**Fundamentos de teste**

Técnica de testes – Como testar

Análise do valor limite

Limites são áreas mais propensas a erros

Limites em partições de equivalência são seus valores mínimos e máximos.



Lembrar que devem ser testados valores mínimos e máximos

Exemplo:

Um campo aceita valores de ano entre 1860 a 1960. Quais os limites deveriam ser testados?

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

63

**Fundamentos de teste**

Técnica de testes – Como testar

Análise do valor limite

Limites são áreas mais propensas a erros

Limites em partições de equivalência são seus valores mínimos e máximos.

Lembrar que devem ser testados valores mínimos e máximos



Exemplo:

Um campo aceita valores de ano entre 1860 a 1960. Quais os limites deveriam ser testados?

1859, 1860, 1960, 1961

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

64

**Fundamentos de teste**

Acabou?
Ainda não!

Dados de testes???

Um assunto Ainda a ser abordado

Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

65

**Dúvidas?**

- Ricardo Ajax
 - Ricardoajax@unb.br



Disciplina206580 – Ver & Val – Prof. Ricardo Ajax (1º/2013)

66