Universidade de Brasília
Faculdade Gama
Engenharia de Software

Disciplina: **206580 – Testes de Software**

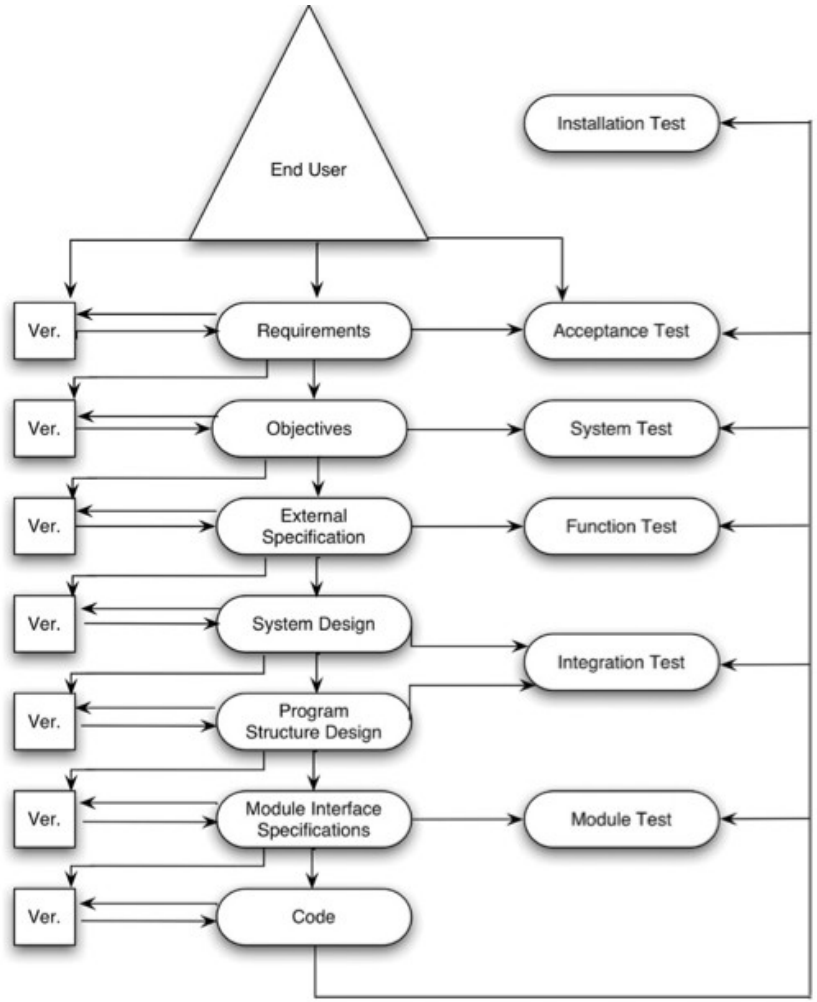
Testes Funcionais, de Sistema e de Aceitação

Elaine Venson
elainevenson@unb.br

Níveis de Teste

1. Testes de unidade
2. Testes de integração
3. **Testes funcionais**
4. **Testes de sistema**
5. **Testes de aceitação**

Níveis de Teste



[Myers]

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***

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
  - Error-guessing

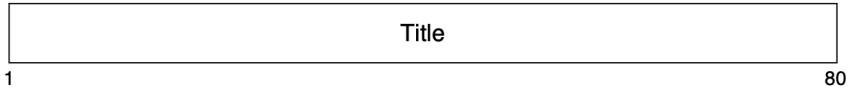
Testes Funcionais

- **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

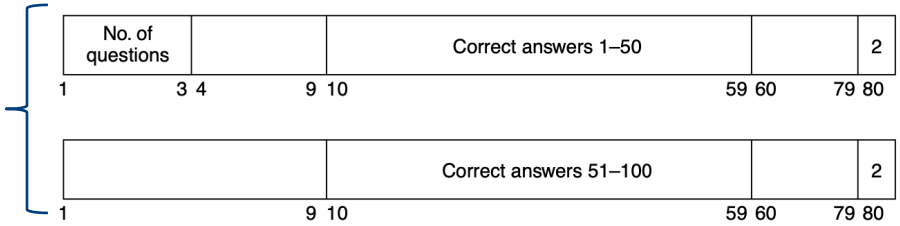
Exemplo: MTEST

- MTEST: programa que corrige provas de múltipla-escolha.
- Entrada: arquivo com linhas de 80 caracteres

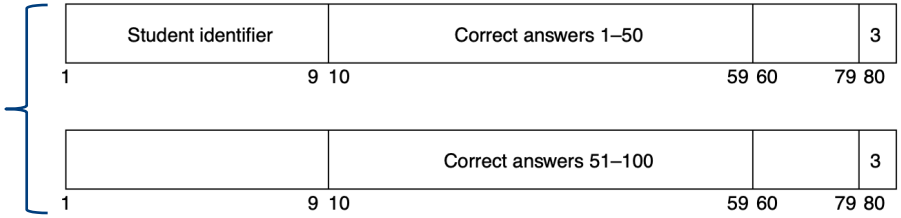
1º. registro:



2º. conjunto de registros:



3º. conjunto de registros:



Número máximo de estudantes: 200

Exemplo: MTEST

- Saídas:

1. Relatório, ordenado pela identificação do estudante, apresentando a nota de cada estudante (percentual de respostas corretas) e classificação (*rank*).
2. Relatório similar, ordenado por nota.
3. Relatório indicando a média, a mediana e o desvio padrão das notas.
4. Relatório, ordenado por número de questão, apresentando o percentual de estudantes que responderam cada questão corretamente.

Exemplo: MTEST

- Passos seguindo Análise de Valor Limite:

1. Ler a especificação procurando por **condições de entrada** e identificar casos de teste:

Primeiro registro:

1. Arquivo vazio
2. Arquivo sem registro de título
3. Título com 1 caractere
4. Título com 80 caracteres

Segundo registro

5. Prova com 1 questão
6. Prova com 50 questões
7. Prova com 51 questões
8. Prova com 999 questões
9. Prova com 0 questões
10. Valor não numérico no campo do número de questões
11. Arquivo sem registro de questões após registro de título
12. Quantidade de questões maior que o número de questões especificado
13. Quantidade de questões menor que o número de questões especificado

Exemplo: MTEST

- Passos seguindo Análise de Valor Limite:

1. Ler a especificação procurando por **condições de entrada** e identificar casos de teste (cont):

Terceiro registro:

14. 0 estudantes
15. 1 estudante
16. 200 estudantes
17. 201 estudantes
18. Um estudante com uma resposta, mas há duas questões
19. O estudante acima é o primeiro estudante do arquivo
20. O estudante acima é o último estudante do arquivo
21. Um estudante tem duas respostas, mas só há uma questão
22. O estudante acima é o primeiro estudante do arquivo
23. O estudante acima é o último estudante do arquivo

Exemplo: MTEST

- Passos seguindo Análise de Valor Limite:

2. Ler a especificação procurando por **condições de saída** e identificar casos de teste:

   Relatórios 1 e 2:

   24. Todos os estudantes recebem a mesma nota
   25. Todos os estudantes recebem notas diferentes
   26. Alguns, mas não todos os estudantes recebem a mesma nota
   27. Um estudante recebe nota 0
   28. Um estudante recebe nota 10
   29. Um estudante tem o menor identificador possível
   30. Um estudante tem o maior identificador possível
   31. O número de estudantes é tal que o relatório cabe em uma página
   32. O número de estudantes é tal que apenas 1 estudante caia na segunda página

Exemplo: MTEST

- Passos seguindo Análise de Valor Limite:

2. Ler a especificação procurando por **condições de saída** e identificar casos de teste (cont):

Relatório 3 (média, mediana e desvio padrão):

33. A média é máxima (todos estudantes com nota máxima)
34. A média é zero (todos estudantes com nota zero)
35. O desvio padrão é máximo (um estudante com 0 e todos os outros com nota máxima)
36. O desvio padrão é zero (todos os estudantes recebem a mesma nota)

Relatório 4:

37. Todos os estudantes respondem a questão 1 corretamente
38. Todos os estudantes respondem a questão 1 incorretamente
39. Todos os estudantes respondem a última questão corretamente
40. Todos os estudantes respondem a última questão incorretamente
41. O número de questões é tal que o relatório cabe em uma página
42. O número de questões é tal que apenas uma questão caia na segunda página

Testes de Sistema

- O **software é apenas um elemento de um sistema** mais amplo
- O *teste de sistema* envolve uma série de diferentes testes, cujo propósito primordial é por completamente à prova o sistema
- 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

Testes de Sistema

- Não há método específico para derivação de casos de teste neste nível
- A elaboração de bons casos de teste requer mais criatividade, inteligência e experiência do que para desenvolver o próprio sistema
- Em geral são aplicadas categorias ou tipos de teste como:
  - Teste de volume
  - Teste de stress
  - Teste de performance
  - Teste de instalação
  - ...

Teste de Aceitação

- É o teste realizado antes da disponibilização do software
- Normalmente realizado pelo cliente ou usuários finais
  - Desenvolvedores cuidadosos irão realizar esse tipo de teste durante o desenvolvimento e antes de realizar a entrega para o usuário/cliente
- 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

Planejamento de Testes

- Por que planejar?
  - As atividades de teste são cruciais em projetos e demandam recursos para sua realização
  - Testar um sistema de grande porte significa escrever, executar e verificar milhares de casos de teste, corrigir milhares de defeitos com o envolvimento de um grande número de pessoas
  - A utilização de recursos sem planejamento pode levar ao desperdício e inabilidade de avaliar a situação das atividades de teste antes da entrega

O Plano de Testes

- Tem como objetivo descrever como os testes serão realizados, recursos e cronograma necessários
- É um documento evolutivo, acompanha as mudanças do projeto
- Deve conter informações sobre o software a ser testado, objetivos e riscos dos testes e tipos de testes que serão realizados
- Informações sobre o status dos testes serão baseadas no plano

Componentes de um Plano de Testes

1. **Objetivos**: os objetivos de cada fase do teste devem ser definidos
2. **Critério de conclusão**: definição dos critérios que especificam quando cada fase de teste será julgada completa
3. **Cronograma**: para cada fase. Deve indicar, por exemplo, quando os casos de teste serão especificados
4. **Responsabilidades**: quem irá projetar, especificar, executar e verificar os casos de teste, além de definir quem irá corrigir os defeitos

Componentes de um Plano de Testes

5. **Padrões e bibliotecas de casos de teste**: grandes projetos exigem formas sistemáticas de identificar, escrever e armazenar casos de teste
6. **Ferramentas**: identificação, plano de aquisição ou desenvolvimento, como serão utilizadas, quando são necessárias
7. **Tempo computacional**: hardware e dispositivos necessários para o teste
8. **Configuração de hardware**: se configurações específicas de hardware e dispositivos são necessárias, como serão obtidas e quando serão utilizadas
9. **Integração**: processo e ordem de integração dos components (top-down, bottom-up etc)

Componentes de um Plano de Testes

10. **Procedimentos de acompanhamento dos testes**: como o progresso dos testes será acompanhado
11. **Procedimentos de depuração**: mecanismos para reportar os erros, progresso da correção, integração do código corrigido
12. **Teste de regressão**: plano para testes de regressão (quem, como, quando) se necessário

Critérios de Conclusão de Testes

- Quando parar de testar?
  - Quando o tempo acabar?
  - Quando todos os casos de teste passarem?

Critérios de Conclusão de Testes

- Categorias de critérios de conclusão:

1. Baseados nos métodos de derivação de casos de teste

   - Ex.: Os casos de teste são derivados a partir de (1) satisfação do critério de cobertura de múltiplas condições e (2) análise de valor limite nos testes de unidade e o resultado dos casos de teste falham eventualmente.

   - Problemas: aplica-se a fases específicas do teste, é uma métrica subjetiva e força o uso dos métodos assinalados

Critérios de Conclusão de Testes

- Categorias de critérios de conclusão:

  2. Baseados na quantidade de defeitos identificados

     - Ex.: Testar até identificar 3 defeitos por unidade
     - Problema 1: Como obter o número de defeitos esperados?
       - Estimar o total de defeitos no software, estimar uma porcentagem a ser identificada por meio dos testes, estimar as frações de defeitos originadas em etapas determinadas do desenvolvimento e em que ponto esses defeitos devem ser identificados
     - Problema 2: Superestimar a quantidade de defeitos

Critérios de Conclusão de Testes

- Categorias de critérios de conclusão:

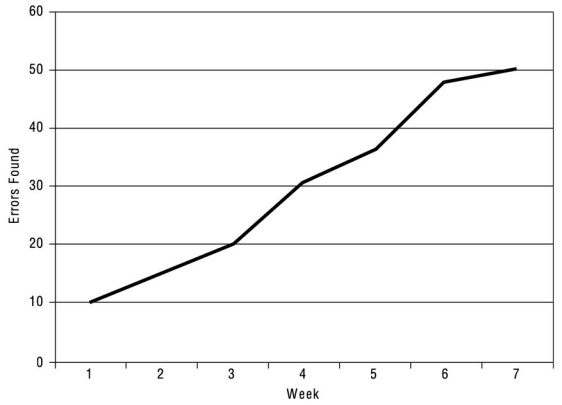
  3. Baseados na curva de defeitos

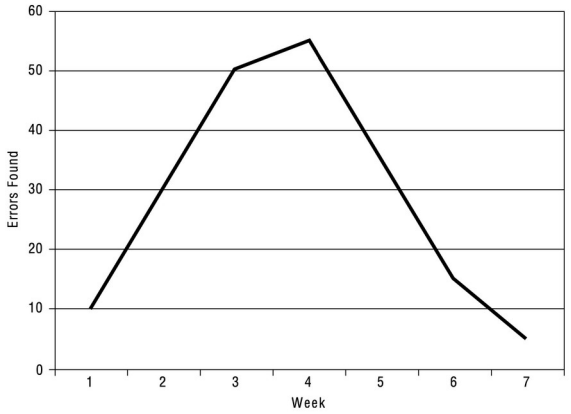
     - Plotar os dados de defeitos encontrados ao longo do tempo e examinar a curva

       - Quando a curva está no pico é imprudente parar

       - Quando a curva está em queda indica que o teste está chegando ao fim

     - Problemas: exige análise, bom julgamento e intuição





Critérios de Conclusão de Testes

- O melhor critério talvez seja a combinação das três categorias:
  - Para o teste de unidade o melhor critério pode ser o primeiro: encerrar os testes quando houver uma cobertura de casos de teste que falharam em algum momento.
  - Para os testes funcionais e de sistema os critérios das categorias 2 e 3 podem ser mais adequados: encerrar os testes quando um número alvo de defeitos foi identificado e usar as curvas de defeitos para analisar o momento quando o teste passa a ser tornar improdutivo (poucos defeitos encontrados).

Referências

- Myers, G. J. et al, The Art of Software Testing. Ed. John Wiley & Sons, 2012. (Capítulos 4 e 6)