ULBRA	**Universidade Luterana do Brasil** **Faculdade de Informática** **Disciplina de Engenharia de Software** **Professor Luís Fernando Garcia – www.garcia.pro.br**

TESTE DE SOFTWARE (Versão 2.0)

9

Teste de Software – Importância

- **Dependência** dos sistemas de software
- Testes como fator de garantia de qualidade - CRUCIAL para garantia da QUALIDADE

- Custos relacionados
  - 2002/USA – 60 bilhões relacionados a falhas de software
  - 2002/USA – 22 bilhões poderiam ser ecomizados caso o processo de teste fosse levado mais a sério
  - Em alguns casos = 40% de todo o custo do projeto
  - Em alguns casos de sistemas "especiais" = 3 a 5 vezes mais que as outras fases juntas

- **Erros clássicos**

  - Estação climática de marte
    - Objetivo – enviar sinais à terra
    - Desastre – chocou-se com o planeta marte
    - Motivo – BUG!
    - Prejuízo – US$ 165 milhões
  - Airbus A320
    - Desastre – USS Vicennes derrubou um A320 em 1988
    - Motivo – BUG! – confundiu um A320 com um F-14
    - Prejuízo – 290 mortes
  - Máquinas de radiação
  - Sistema de ambulância de Londres 1992
  - Airbus A300 – queda em 1994 – 264 mortes
  - Fogo amigo na guerra das Malvinas
  - Trem parado no meio do nada – erro ao detectar a estação
  - Míssil Scud na guerra do golfo – erro em tempo real de .36s – 30 mortos

Expectativas

- Programas feitos com bastante cuidado
  - 5 falhas / 1000 LOC
  - Windows XP → 45 milhões LOC → 225.000 erros!

Teste de Software – Definição

Processo de executar um programa com o objetivo de revelar a presença de **erros** ...
- Verificação imcompleta
- Não garante a inexistência de erros no programa
- Caros

"Teste consiste na verificação dinâmica do funcionamento de um programa em um conjunto finito de casos de teste, cuidadosamente selecionado dentro de um domínio infinito de entradas, contra seu funcionamento esperado."
- Dinâmico – Execução
- Finito – Existem muitos casos de teste
- Selecionado – Técnicas diferem na seleção
- Esperado – Funcionamento deve ser verificado

Terminologia

- Fault (falta) → Causa de um mal funcionamento.
- Failure (falha) → Efeito observável não desejável.
- Erro → Qualquer diferença entre um valor obtido, independente da sua forma de obtenção, e o valor teoricamente correto.
- Exceção → Evento que causa a suspensão da operação normal de um programa.
- Anomalia → Qualquer coisa observada na operação ou documentação de um programa que não está de acordo com as expectativas, baseado em documentos ou em outros programas.

- Verificação → Estamos construindo certo o produto?
- Validação → Estamos construindo o produto certo?
- Oráculo → Entidade responsável pela determinação dos resultados esperados em um teste

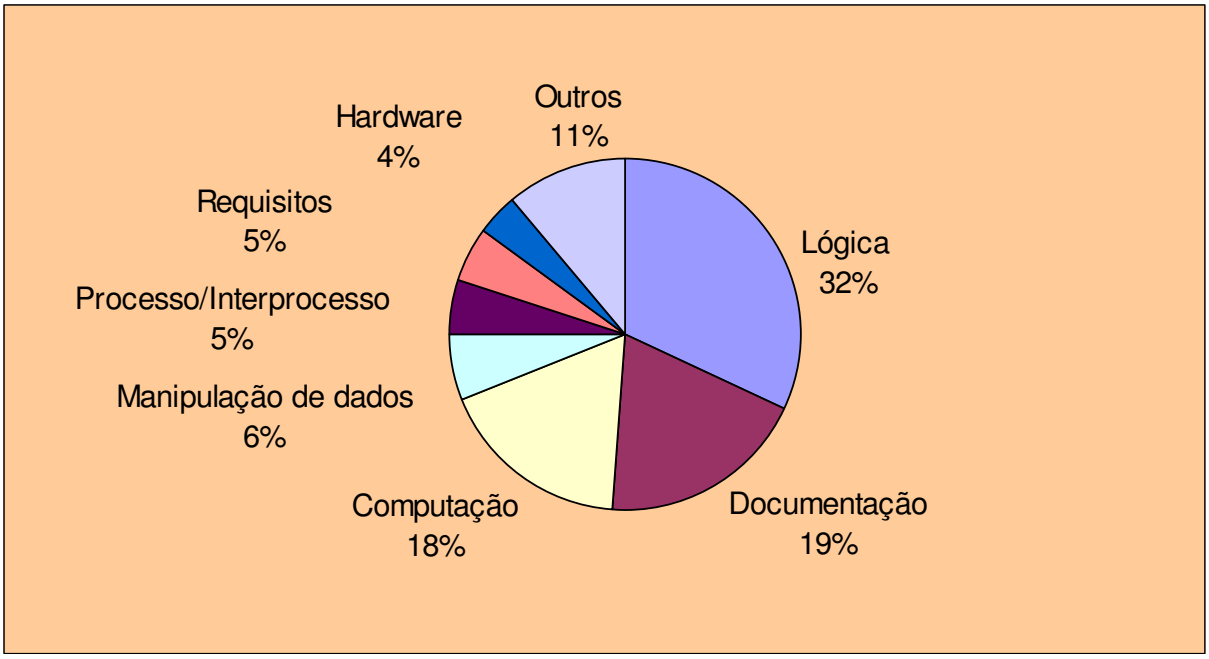
Prováveis Defeitos no processo de desenvolvimento de software

- A maior parte é de origem humana – humanos não são 100% perfeitos
- Podem começar a aparecer desde a especificação de requisitos
    - Especificação errada – não corresponde a necessidade do cliente
    - Especificação errada – requisitos impossíveis de implementar
- São gerados na comunicação e na transformação de informações
- Continuam presentes nos diversos produtos de software produzidos e liberados (10 defeitos a cada 1000 linhas de código)
- A maioria encontra-se em partes do código raramente executadas
- Principal causa: tradução incorreta de informações
- Quanto antes a presença do defeito for revelada, menor o custo de correção do efeito e maior a probabilidade de corrigi-lo corretamente
- Solução: introduzir atividades de VV&T ao longo de todo o ciclo de desenvolvimento

Tipos de Falhas

- Algoritmo – algoritmo não produz a saída esperada
  - Esquecer de inicializar variáveis
  - Teste de condições erradas – desvios de código errado
- Sintaxe – linguagens de programação usadas indevidamente
  - Esquecer de ponto-e-vírgula no final da linha em Pascal
- Computação e precisão
  - Erros em fórmulas matemáticas
  - Utilização errada de precisão – casas decimais e etc...
- Documentação
  - Código e documentação inconsistentes
- Overload – sobrecarga
  - Estouro de estruturas de dados – como vetores e matrizes
- Falha de Limite e Capacidade
  - Capacidade de tratamento = 32 dispositivos; Mais de 32 = erro
- Coordenação e Timing
  - Sistemas Distribuídos e de Tempo Real
  - Sistemas bancárias – controle de transações
- Desempenho
  - Velocidade do sistema não está de acordo com os requisitos
- Recuperação
  - Falhas em decorrência de eventos externos
  - Eventos como falta de luz
  - Tolerância a falhas
- Hardware
  - Normalmente relacionadas a DRIVERS de dispositivos
- Falhas de Padronização e procedimentos
  - Padronização de codificação – entendimento do código

Estatísticas de Falhas – uma divisão da HP - 1997



Princípios de teste

- não planeje o teste assumindo que o programa está correto
- um bom caso de teste é aquele que tem alta probabilidade de encontrar erro ainda não descoberto
- caso de teste bem sucedido é aquele que detecta erro ainda não descoberto
- a probabilidade de existência de mais erros numa parte do programa é proporcional ao número de erros já descoberto na mesma
- teste deve ser feito por outra pessoa que não o autor do programa
- dados de teste devem ser definidos para dados inválidos e não-esperados
- determinar SEMPRE os resultados esperados
- verificar cuidadosamente os resultados de cada teste
- nunca jogue fora casos de teste, a não ser que esteja jogando fora também seu programa

Etapas do processo de Teste

Não é / não deve ser uma FASE do processo de desenvolvimento de software
Deve ser feita em paralelo com as outras FASES do processo

Analise ←→ Teste
Projeto ←→ Teste
Codificação ←→ Teste

- Planejamento – objetivos
- **Projeto de casos de teste**
- Execução do programa com os casos de teste
- Análise dos resultados

Casos de teste

- Dados usados para testar um software
- Especificação de uma entrada para o programa e a saída esperada correspondente
- Um bom caso de teste tem alta probabilidade de revelar um erro ainda não descoberto

Projeto de casos de teste

- Pode ser tão ou mais difícil quanto o projeto do produto testado
- Poucos profissionais gostam ou sabem realizar projetos de teste

- Deve ser finito
- Custo de aplicação tem que ser razoável

Testes – Classificação

- Análise Estática
    - Análise do código-fonte sem executá-lo
    - Inspeções – entender o programa para descobrir erros
    - Walkthroughs – execução simulada do programa
- Análise Dinâmica
- Teste Simbólico
    - Executar o programa com dados simbólicos
    - Predicados de caminhos – utilização de grafos
- Verificação Formal
    - Especificação formal do programa

- Provador de teoremas
- Difíceis
- Caros
- Demorado

Técnicas e critérios de teste

Teste funcional – caixa preta – requisitos funcionais do software
Teste Estrutural – caixa branca – estrutura interna do programa
Técnica baseada em erros – erros mais freqüentes durante o processo de desenvolvimento

Teste de unidade – identificar erros de cada módulo separadamente

- Concentra-se no módulo
- Técnicas de teste estrutural e baseada em erros
- Pode ser realizado em paralelo
- Usam
  - Driver – Espécie de "programa principal" que aciona o módulo
  - Módulo a ser testado
  - Stub – Módulos "substitutos" com o mínimo de manipulação de dados
    - Módulos chamados pelos módulos testados

- Características avaliadas
  - Interface
    - Dados entram e saem corretamente?
    - Compatibilidade de parâmetros?
  - Operações sobre variáveis
    - Cálculos incorretos?
    - Overflow ou underflow?
  - Caminhos de execução importantes
  - Caminhos de atendimento a erros
  - Condições de contorno

Teste de integração – identificar erros associados as interfaces entre os módulos de software

- Necessário para verificar erros ocorridos quando da junção dos módulos
- Abordagens
  - Top-down
  - Botton-Up
  - Mista

Teste de validação do sistema – verificar se as funções estão de acordo com a especificação

- Demonstrar a conformidade com a especificação
- Verificar se a documentação está correta
- Abordagens
  - Teste Alfa – realizado pelo usuário no ambiente do desenvolvedor
  - Teste Beta – realizado pelo usuário em seu próprio ambiente

Teste de Sistema

- Considera o software dentro do seu ambiente mais amplo (HW, SW, pessoas)
- Teste de segurança
- Teste de estresse – condições anormais de volume e recursos
- Teste de desempenho – tempo de resposta

Ferramentas de Teste

Uso de ferramentas automatizadas de teste que apóiam o processo de teste
Reduzem as falhas introduzidas pela intervenção humana
Aumento de produtividade
Facilitam estudos comparativos entre critérios

Possíveis conclusões

- A atividade de teste é fundamental no processo de desenvolvimento de software
  - Qualidade do produto
- Alto custo da atividade de teste
- Desenvolvimento e aplicação de técnicas e critérios de teste
- Desenvolvimento e utilização de ferramentas de teste
- Estudos teóricos e empíricos para comparar os diversos critérios

Questionário

1. Qual a importância de se testar um software?
2. Se uma empresa desenvolver um software de forma cuidadosa, utilizandos todas as técnicas de engenharia de software para as fases de análise, projeto e codificação, pode-se pular a fase de testes? Por quê?
3. Cite três motivos que podem causar erros em um software.
4. Quando se diz que um teste é bem sucedido?
5. Suponha um programa com 1 milhão de linhas de código desenvolvido por 10 programadores em um mês. Suponha que a fase de testes não encontrou nenhum erro. Pode-se considerar que o programa está livre de erros? Ou pode-se supor que os testes não foram bem feitos? Por quê?
6. Cite um exemplo de falha na sintaxe.
7. Por que a falha de documentação é grave? Dê um exemplo.
8. Qual a diferença entre a falha de overload e a falha de limite?
9. O que é falha de recuperação?
10. O que é falha de padronização? Pode ser considerada uma falha grave ou leve?
11. Qual é a diferença entre os testes de caixa preta e os testes de caixa branca?
12. Cite pelo menos 3 tipos de erros que podem ser encontrados em testes do tipo caixa preta.
13. Testes do tipo caixa preta podem ser utilizados para descobrir em que local do código está o erro?