UFU- Universidade Federal de Uberlândia
FACOM – Faculdade de Computação

V, V & T

Validação, Verificação e Teste

Slides adaptados do Prof. Dr. Tiago Silva da Silva (Unifesp)

Objetivo

- Importância de “testar” software
- Introduzir conceitos de V, V & T
- Técnicas de Teste de software

Relembrando...

- Qual o objetivo da Engenharia de Software?

Relembrando...

- Qual o objetivo da Engenharia de Software?
  - Melhorar a qualidade do software

Relembrando...

- Qual o objetivo da Engenharia de Software?
  - Melhorar a qualidade do software
  - Qualidade do produto e qualidade do processo

Relembrando...

- Qual o objetivo da Engenharia de Software?
  - Melhorar a qualidade do software
    - Qualidade do produto e qualidade do processo
- Como garantir a qualidade?

Relembrando...

- Qual o objetivo da Engenharia de Software?
  - Melhorar a qualidade do software
    - Qualidade do produto e qualidade do processo
- Como garantir a qualidade?
  - Certificar que o software atende sua especificação.

Validação, Verificação e Teste de Software (V, V & T)

Introdução

Importância

- Errar (ou enganar-se) é humano - somos limitados

Introdução

Importância

- Qual é a porcentagem do tempo e do custo total de desenvolvimento de software que é, em geral, gasto com teste?

Introdução

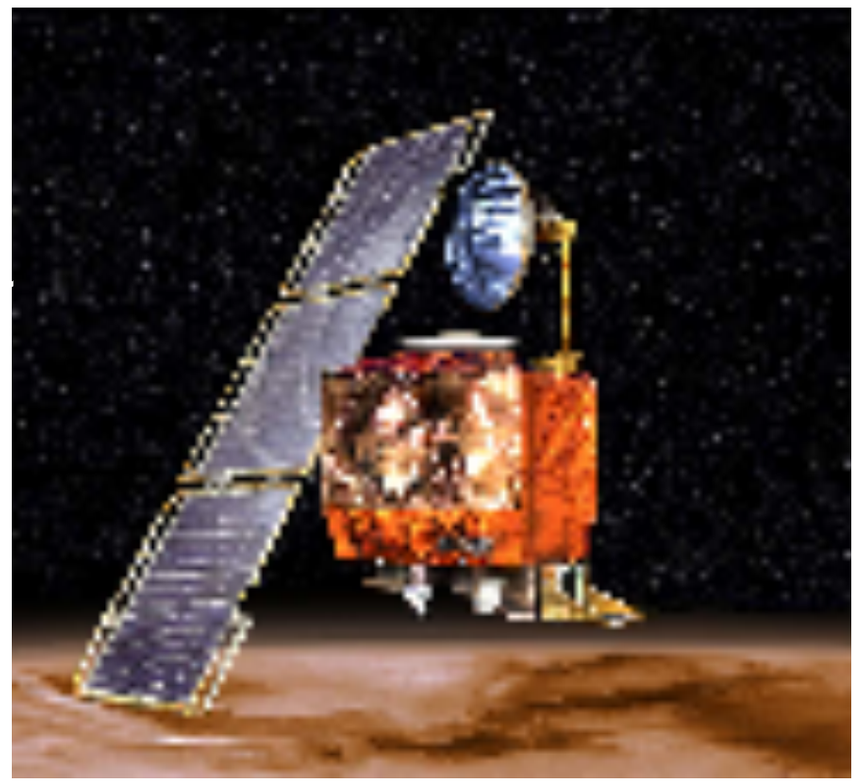
Importância

- Qual é a porcentagem do tempo e do custo total de desenvolvimento de software que é, em geral, gasto com teste?
- R: **50% do tempo e + de 50% do custo**

Introdução

Importância

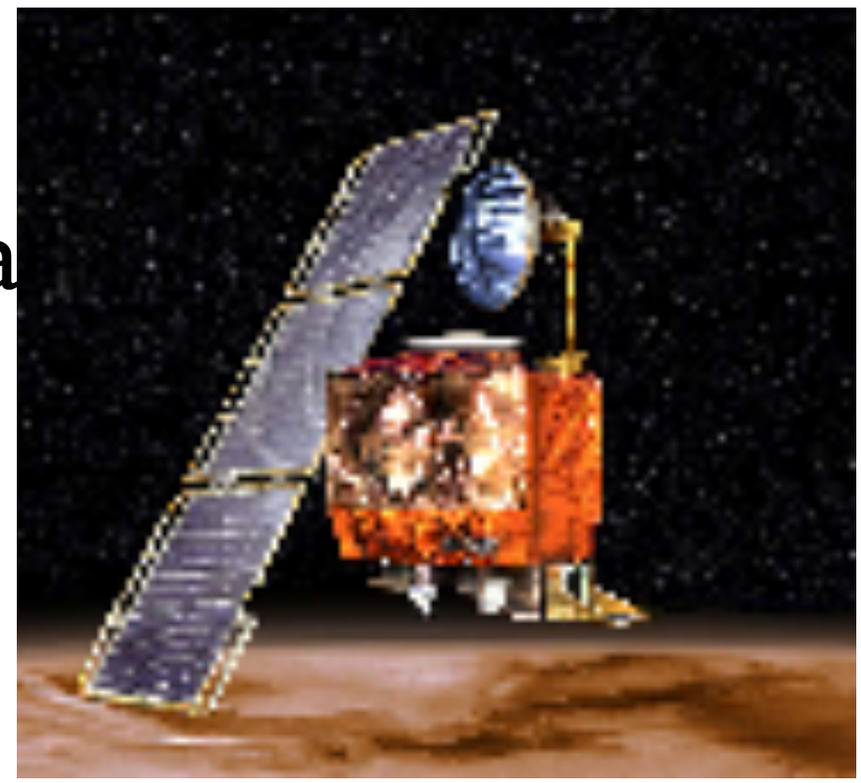
- 1999: Mars climate orbiter - propósito: mandar sinais da sonda Mars Polar
- Desastre: caiu no planeta em vez de atingir órbita



Introdução

Importância

- 1999: Mars climate orbiter - propósito: mandar sinais da sonda Mars Polar
- Desastre: caiu no planeta em vez de atingir órbita
- Por que: bug - falha em converter unidades de medida
- US$125M



Introdução

Importância

- 1998: US Vincennes derrubou um Airbus 320
- Confundiu com um F-14: 290 pessoas morreram

Introdução

Importância

- 1998: US Vincennes derrubou um Airbus 320
- Confundiu com um F-14: 290 pessoas morreram
- Por que: bug - saída estranha dada pelo software de rastreamento

Introdução

Importância

- THERAC-25, máquina de tratamento de radiação controlada por computador
- 1986: dois pacientes com câncer do Centro de Câncer do Texas receberam doses fatais de radiação

Introdução

Importância

- THERAC-25, máquina de tratamento de radiação controlada por computador
- 1986: dois pacientes com câncer do Centro de Câncer do Texas receberam doses fatais de radiação
- Por que: bug - falha em condição de concorrência (tarefas concorrentes não coordenadas corretamente)

Introdução

Importância

- Serviço de Despacho de Ambulâncias de Londres
- Propósito: automatizar processos manuais associados ao serviço de ambulâncias do Reino Unido
- Falhou em 26 e 27 de Novembro de 1992

Introdução

Importância

- Carga aumentou, emergências acumularam - sistema fez alocações incorretas
  - mais de uma ambulância no mesmo incidente
    - veículo mais próximo não foi alocado
- No dia seguinte...
  - mais ou menos 46 mortes poderiam ter sido evitadas

Introdução

Importância

- Mais:
  - Navio britânico H.M.S. Sheffield: afundou - sistema de alerta de radar permitiu que o míssil o antigisse
  - Avião neo-zelandês trombou em montanha
  - Comando de Defesa Aeroespacial Americano: EUA sob ataque de mísseis - software gerou sinais incorretos
  - Cápsula espacial Gemini V errou local de pouso por 100 milhas: software não considerou o movimento de rotação da Terra ao redor do sol

Introdução

Questões importantes

- A maior parte dos defeitos é de origem **humana**
- 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

Introdução

Questões importantes

- Psicologia adequada - contribui mais
- Consideração vital - definição de ‘teste’
- Premissa - quase todas as pessoas usam uma definição incorreta dessa palavra
- Causa primária da falta de eficácia

Questões importantes

- Quanto antes defeito revelado - menor custo de correção; maior a probabilidade de corrigi-lo corretamente
- Principal causa: tradução incorreta de informações
- Solução: atividades de VV&T ao longo do ciclo

Introdução

Questões importantes

- Defeitos não revelados -

Introdução

Questões importantes

- Defeitos não revelados - falhas no cliente

Introdução

Questões importantes

- Defeitos não revelados - falhas no cliente
- Defeitos corrigidos durante manutenção -

Introdução

Questões importantes

- Defeitos não revelados - falhas no cliente
- Defeitos corrigidos durante manutenção - Alto custo

Objetivos Principais

- Descobrir defeitos em um sistema

Introdução

Objetivos Principais

- Descobrir defeitos em um sistema
- Avaliar se o sistema é útil e usável ou não em uma situação operacional

Introdução

Objetivos Principais

- Descobrir defeitos em um sistema
- Avaliar se o sistema é útil e usável ou não em uma situação operacional
- Nível de confiabilidade (rigor) varia: função do software; expectativas do usuário; ambiente de mercado

Fundamentos

VV&T

- **Validação**:

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?
- **Verificação**:

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?
- **Verificação**: assegurar a consistência, completude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?
- **Verificação**: assegurar a consistência, completude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software
  - Estamos construindo corretamente o produto?

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?
- **Verificação**: assegurar a consistência, completude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software
  - Estamos construindo corretamente o produto?
- **Teste**:

Fundamentos

VV&T

- **Validação**: assegurar que o produto final corresponda aos requisitos do software
  - Estamos construindo o produto certo?
- **Verificação**: assegurar a consistência, completude e corretude do produto em cada fase e entre fases consecutivas do ciclo de vida do software
  - Estamos construindo corretamente o produto?
- **Teste**: Examina o comportamento do produto através de sua execução (Atividade Dinâmica!)

Fundamentos

Defeitos, Erros e Falhas

Defeito - Erro - Falha

Fundamentos

Defeitos, Erros e Falhas

Defeito - Erro - Falha

- **Defeito**: deficiência mecânica ou algorítmica que, se ativada, pode levar a uma falha

Fundamentos

Defeitos, Erros e Falhas

Defeito - Erro - Falha

- **Defeito**: deficiência mecânica ou algorítmica que, se ativada, pode levar a uma falha
- **Erro**: item de informação ou estado de execução inconsistente

Fundamentos

Defeitos, Erros e Falhas

Defeito - Erro - Falha

- **Defeito**: deficiência mecânica ou algorítmica que, se ativada, pode levar a uma falha
- **Erro**: item de informação ou estado de execução inconsistente
- **Falha**: evento notável em que o sistema viola suas especificações - saída obtida != saída esperada

Teste de Software

Definições

- Definição de teste de software mais apropriada?
  - "O processo de demonstrar que defeitos não estão presentes."
  - "O propósito de testar é mostrar que um programa realiza suas funções corretamente."
  - "Teste é o processo de estabelecer confiança de que um programa faz o que deve fazer."

Teste de Software

Definições

- Definição de teste de software mais apropriada:
  - Processo de executar um programa com o objetivo de revelar a presença de defeitos.
  - Premissa correta: quase impossível não haver defeitos
  - Teste de sucesso?

Teste de Software

Definições

- Quando um teste é considerado de sucesso?

Teste de Software

Definições

- Quando um teste é considerado de sucesso?
- R: Quando ele falha.

Teste de Software

Definições

- Quando um teste é considerado de sucesso?
  - R: Quando ele falha.
  - R: Quando você esperava que ele não falhasse e ele falha; ou quando você esperava que ele falhasse e ele não falha.

Teste de Software

Definições

- Domínio de entrada: $D(P)$ - todos possíveis valores de entrada para *P*
- Exemplo: $P(x, y), y \geq 0 \rightarrow x^y$
  - (x, y)
- Domínio de saída – todos os possíveis resultados

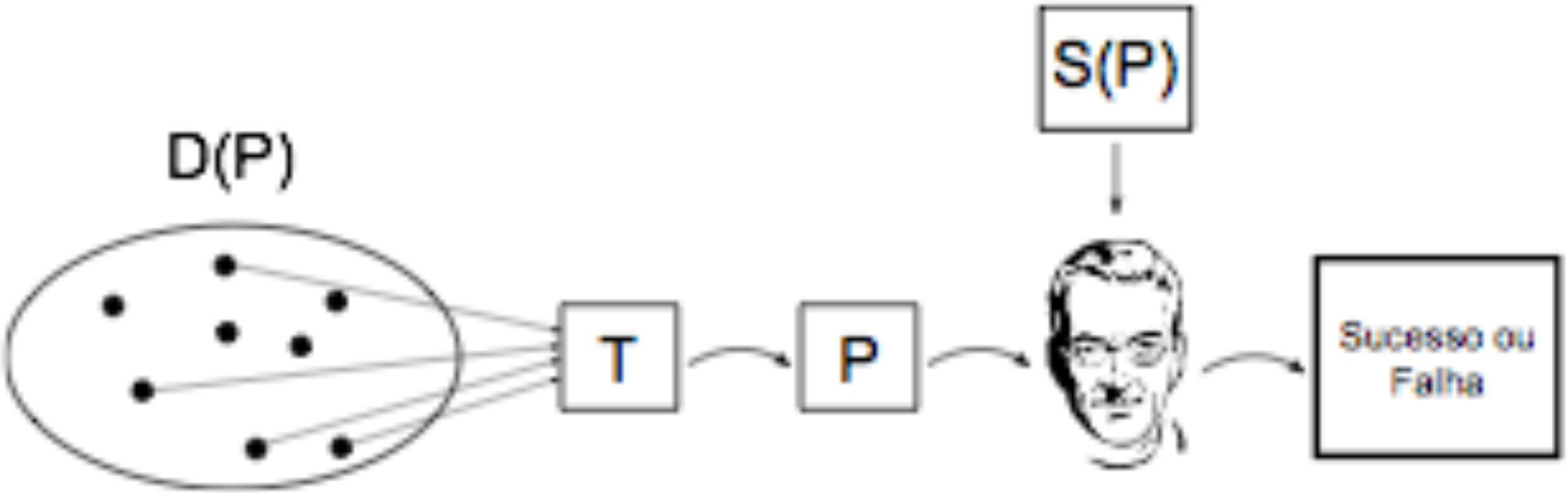
Teste de Software

Definições

- Dado de teste: $d \in D$
- Caso de teste: $(d, P(d))$
  - Exemplos:
    - <(2, 3, 8)>, <(4, 3), 64>, <(3, -1), Erro>
- Como saber a saída esperada?
- Especificação do programa: $S(P)$
- Análise: a cargo do testador - oráculo

Teste de Software

Cenário Típico



Fases da atividade de teste

- A atividade de teste é complexa!
  - A utilização de um algoritmo incorreto para computar o valor das mensalidades a serem pagas para um empréstimo
  - A não-utilização de uma política de segurança em alguma funcionalidade do software

Fases da atividade de teste

- A utilização de um algoritmo incorreto para computar o valor das mensalidades a serem pagas para um empréstimo
  - Confinado a uma função implementada de forma incorreta.
- A não-utilização de uma política de segurança em alguma funcionalidade do software
  - Mesmo que exista uma certa política de segurança implementada de forma correta, é preciso verificar se todos os pontos nos quais essa política deveria ser aplicada fazem-no de maneira correta.

Teste de Software

Fases da atividade de teste

- Por isso, a atividade de teste é dividida em fases com objetivos distintos:
  - Teste de Unidade
  - Teste de Integração
  - Teste de Sistemas

Teste de Software

Fases da atividade de teste

- Teste de Unidade

Fases da atividade de teste

- Teste de Unidade
  - foco nas menores unidades de um programa; funções, procedimento, métodos ou classes
  - Espera-se que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas, ou simples erros de programação
  - Cada unidade é testada separadamente
  - Pode ser aplicado à medida que ocorre a implementação das unidades e pelo próprio desenvolvedor

Teste de Software

Fases da atividade de teste

- Teste de Integração

Fases da atividade de teste

- Teste de Integração
  - deve ser realizado após serem testadas as unidades individualmente
  - a ênfase é dada na construção da estrutura do sistema
  - À medida que as diversas partes do software são colocadas para trabalhar juntas, é preciso verificar se a interação entre elas funciona de maneira adequada e não leva a erros
  - Tende a ser executado pela própria equipe de desenvolvimento

Teste de Software

Fases da atividade de teste

- Teste de Sistema

Teste de Software

Fases da atividade de teste

- Teste de Sistema
  - depois que o sistema está “completo”, com todas as suas partes integradas
  - Objetivo é verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas
  - Correção, completude e coerência são aspectos a serem explorados
  - Requisitos não funcionais como segurança, performance e robustez
  - Pode-se designar uma equipe independente para realizar o teste de sistemas

Teste de Software

Fases da atividade de teste

- Teste de “Regressão”

Teste de Software

Fases da atividade de teste

- Teste de “Regressão”
  - durante a manutenção
  - A cada modificação efetuada no sistema, após a sua liberação, corre-se o risco de que novos defeitos sejam introduzidos
  - É necessário realizar testes que mostrem que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados funcionam como o esperado E
    - que os requisitos anteriormente testados continuam válidos

Teste de Software

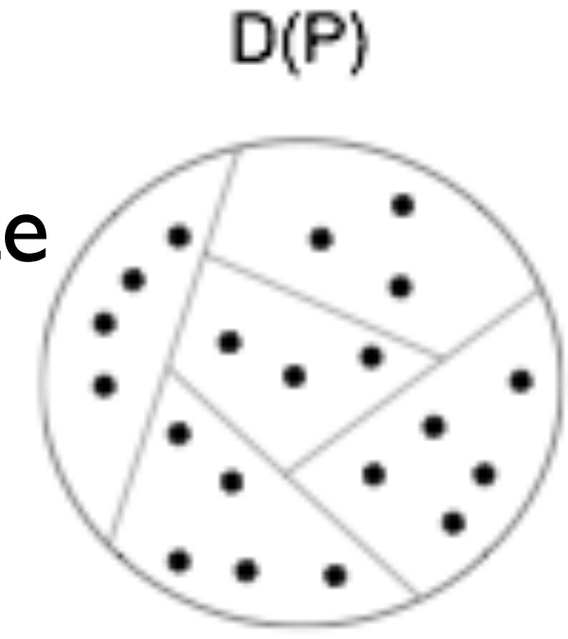
Técnicas e Critérios

- Idealmente - exaurir $D(P)$
- Em geral, infactível: no exemplo - $|D(P)| = 2^n * 2^n$
- Em uma arquitetura de 32 bits - 2^{64} = 18446744073709551616 dados de teste
- Se cada caso de teste executa em 1ms - 5.849.424 séculos

Teste de Software

Técnicas e Critérios

- Ideia - subconjunto de *D(P)* com alta probabilidade de encontrar defeitos - subdomínios de teste
- Exemplo: <(2, -2), Erro> e <(2, -1), Erro> - semelhantes
  - Testar um
- Reduz-se *D(P)*, mas com casos de teste significativos



Teste de Software

Técnicas e Critérios

- Em essência, duas formas: aleatório e partição (subdomínios) - com vantagens e desvantagens
- Teste aleatório
  - Número de casos de teste é selecionado aleatoriamente
- Teste de partição (teste de subdomínios)
  - Procura-se estabelecer subdomínios a serem utilizados e, então, selecionam-se os casos de teste em cada subdomínio

Teste de Software

Técnicas e Critérios

- Teste de subdomínios:
  - definem-se Regras - requisitos de teste (Ex.: dados que passam por determinada estrutura; número positivos/números negativos)
    - definem-se Requisitos: dados que satisfazem requisito estão no mesmo subdomínio
- Critérios de teste **C** - definem diferentes subdomínios

Teste de Software

Técnicas e Critérios

- Técnicas: em geral, três tipos -
  - Funcional (caixa preta)
  - Estrutural (caixa branca)
  - Baseada em defeitos

Dez princípios do Teste de Software

Dez princípios do Teste de Software

1. A saída esperada é parte essencial de um caso de teste
2. Um programador deve evitar testar seu próprio programa
3. Uma empresa deve evitar testar seus próprios programas
4. Inspecione cuidadosamente os resultados dos casos de teste
5. Casos de teste devem ser escritos para as condições de entrada inválidas e inesperadas

Dez princípios do Teste de Software

6. Examinar se um programa não faz aquilo que deveria fazer é metade do trabalho; a outra metade é verificar se o programa faz o que não deveria fazer
7. Evite casos de teste inúteis a não ser que o programa em si seja útil
8. Não planeje uma atividade de teste com a hipótese de que nenhum defeito será encontrado
9. A probabilidade da existência de mais erros em uma parte de um programa é proporcional ao número de defeitos já encontrados naquela parte
10. Teste é uma atividade extremamente criativa e desafiante