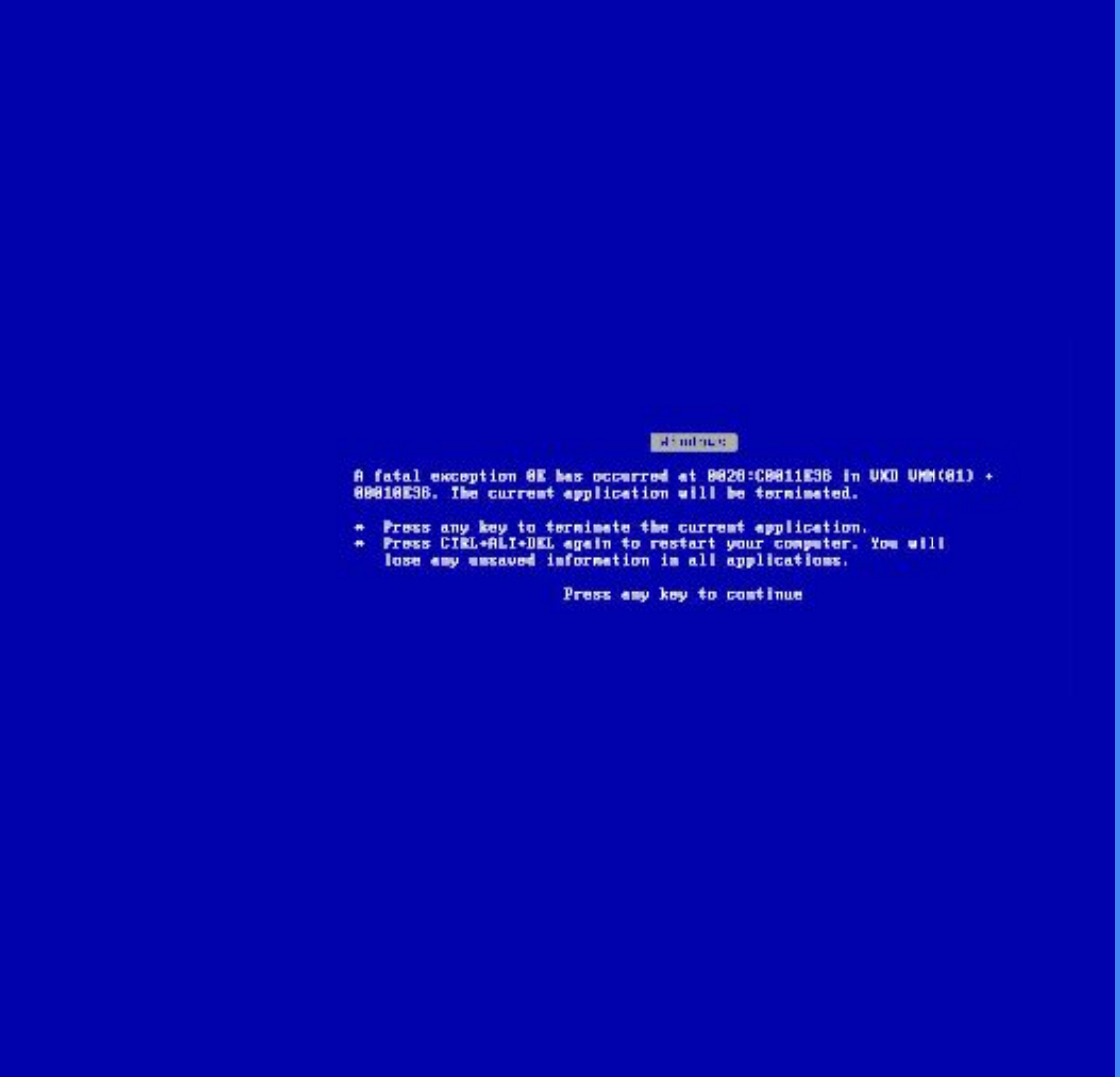


UniCesumar

Projeto, implementação e Teste de Software

Princípios e técnicas de teste de software

Prof. Esp. Dacio F. Machado


LOADING...

Anteriormente em Teste de Software

Testar um software consiste em:

- Verificar se ele atende às expectativas
- Se seu funcionamento é limpo, amigável e correto
- Se ele se enquadra no ambiente para o qual foi projetado

O teste tenta garantir que o programa funcione conforme o esperado, seja seguro, confiável e atenda às necessidades dos usuários

Porque Testar software ?

- Para assegurar que as necessidades dos usuários estejam sendo atendidas.
- Porque é provável que o software possua defeitos.
- Desenvolvedor já alocado para outro projeto teria que resolver
- muitos bugs de projetos anteriores em produção.
- Porque falhas podem custar muito caro.
- Para avaliar a qualidade do software

Muitos fatores podem ser identificados como causas de tais problemas, mas a maioria deles tem uma única origem: **erro humano** "(DELAMARO; MALDONADO; JINO, 2007,).

- Erro: é uma ação humana que produz um resultado incorreto.
- Defeito: A manifestação de um erro no software. Também conhecido como Bug
- Falha: quando o sistema se comporta de forma inesperada devido ao defeito

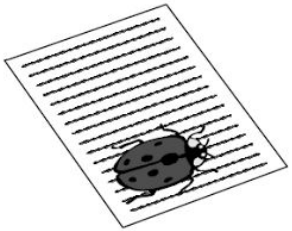
Reflexão

Quando executamos testes em um software, podemos demonstrar a presença de defeitos, mas não podemos provar que eles não existem. Você concorda com isso ?

Aula de Hoje

Princípios e técnicas de teste de software - <u>Teoria</u>

- **Conceitos Básicos em Teste de Software**
- **Limitações teóricas de Teste de Software**
- **Artefatos de Teste de Software**
- **Estratégias de Teste de Software**
- **Conceitos sobre Qualidade de Software**



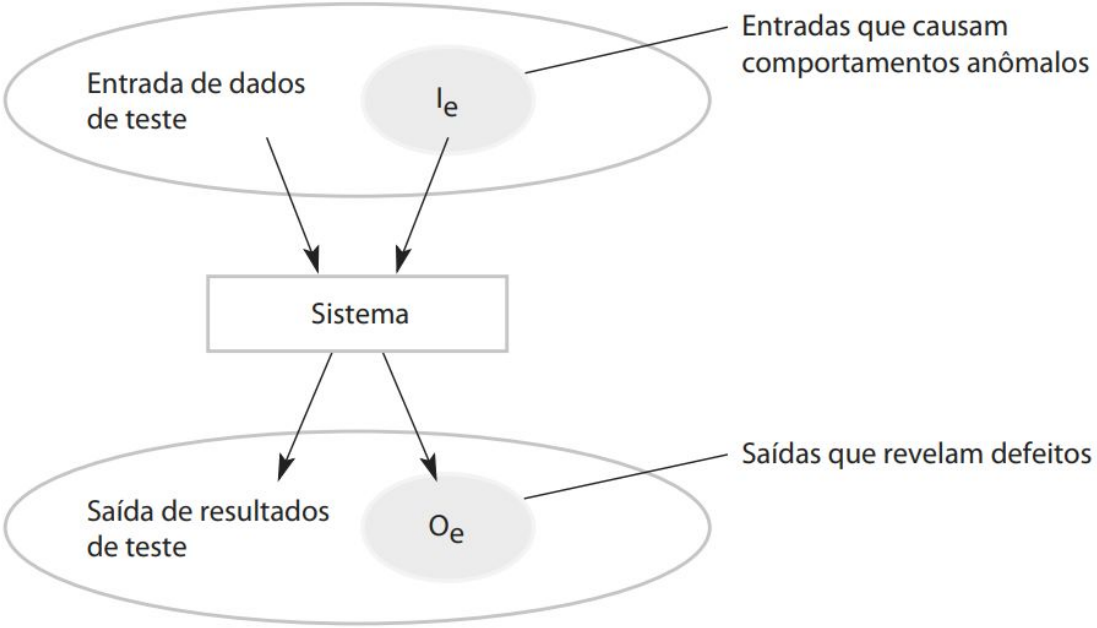
Conceitos Básicos em Teste de Software

Diz-se que um programa P, com domínio de entrada D, é **correto** com respeito a uma especificação S se: $S(d) = P^*(d)$ para qualquer item de dado $d \in D$**.** Ou seja, o comportamento do programa está de acordo com o comportamento esperado para todos os dados de entrada.

Dados dois programas $P1$ *e* $P2$, se: $P1^*(d) = P2^*(d)$ para todo $d \in D$, então $P1$ e $P2$ são considerados ***equivalentes***.

No **teste de software**, pressupõe-se a existência de um **ORÁCULO** – o testador ou algum outro mecanismo – que possa determinar, para qualquer dado $\mathbf{d} \in \mathbf{D}$, se: $S(d) = P^*(d)$ dentro de limites razoáveis de tempo e esforço. Um ***oráculo*** simplesmente decide se os valores de saída apresentados por P estão corretos em relação à especificação S.

Figura 8.1 Um modelo de entrada-saída de teste de programa



Limitações teóricas de Teste de Software

A atividade de teste é permeada por uma série de limitações. Em geral, os seguintes problemas são **indecidíveis**:

- Dados dois programas, determinar se eles são **equivalentes**;
- Dadas duas sequências de comandos (ou caminhos), seja de um mesmo programa ou de programas diferentes, decidir se elas **computam a mesma função**;
- Dado um caminho, verificar se ele é **executável**, ou seja, se existe um dado de entrada $d \in D$ que leva à execução desse caminho

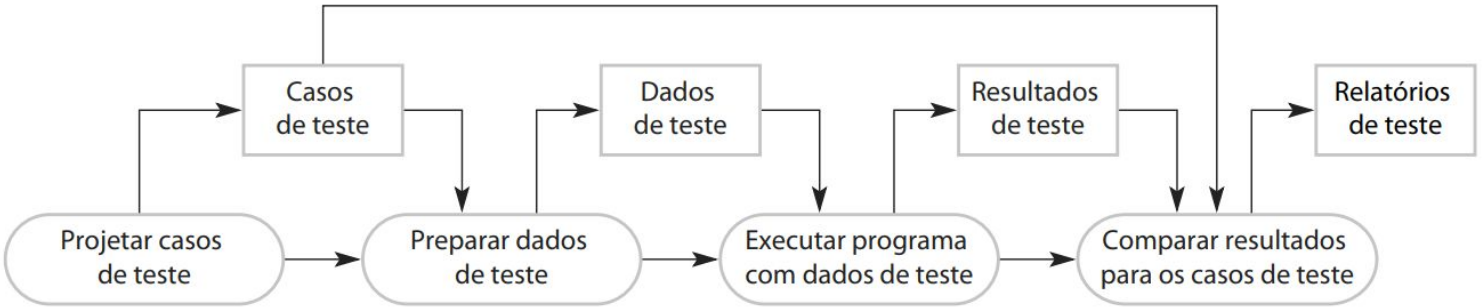
Limitações teóricas de Teste de Software

- Outra limitação fundamental é a chamada **correção coincidente** – o programa pode apresentar, por coincidência, um **resultado correto** para um item específico $d \in D$. Nesse caso, o dado satisfaz um requisito de teste e **não faz o software falhar**, mesmo que o programa esteja incorreto de forma geral

Conceitos Básicos em Teste de Software

Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. Para isso precisamos definir alguns conceitos, papéis e artefatos necessários para estruturar um teste de software da maneira correta.

Figura 8.3 Um modelo do processo de teste de software



Artefatos de Teste	todo o conjunto de documentação gerado pelo processo de teste de software.
Caso de Teste	é composto por um conjunto de entradas, por passos de execução e um resultado esperado
Roteiro de Teste	é composto por um conjunto de casos de teste definidos para uma determinada especificação

Conceitos Básicos em Teste de Software

PLANO DE TESTE: É um documento que descreve a abordagem geral.

CASOS DE TESTE: São documentos que descrevem os cenários específicos que serão testados

ROTEIRO DE TESTE: São documentos que detalham a sequência de etapas que os testadores devem seguir ao executar os casos de teste

RELATÓRIO DE EXECUÇÃO DE TESTES: São documentos que resumem os resultados dos testes executados

Conceitos Básicos em Teste de Software

Às vezes, os dados de teste podem ser gerados automaticamente, mas a geração automática de casos de teste é impossível, pois as pessoas que entendem o propósito do sistema devem ser envolvidas para especificar os resultados esperados.

No entanto, a execução do teste pode ser automatizada.

Os resultados esperados são automaticamente comparados aos resultados previstos,(**oráculo**) por isso não há necessidade de uma pessoa para procurar erros e anomalias na execução dos testes

Entretanto, testes automatizados só encontram defeitos esperados.

Sommerville (2011, p. 147)

Estratégias de Teste de Software

Muitas estratégias de teste de software já foram propostas na literatura. Todas elas fornecem um modelo para o teste e todas têm as seguintes características genéricas:

- Antes de executar um teste eficaz, deve proceder com as revisões técnicas. Fazendo isso, muitos erros serão eliminados antes do começo do teste.
- O teste começa no nível de componente e progride em direção à integração do sistema computacional como um todo

Estratégias de Teste de Software

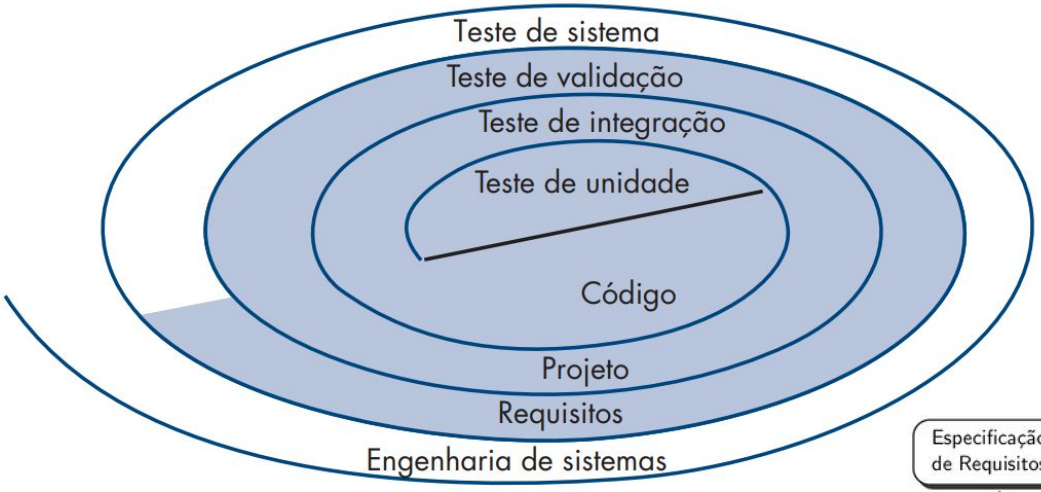
Muitas estratégias de teste de software já foram propostas na literatura. Todas elas fornecem um modelo para o teste e todas têm as seguintes características genéricas:

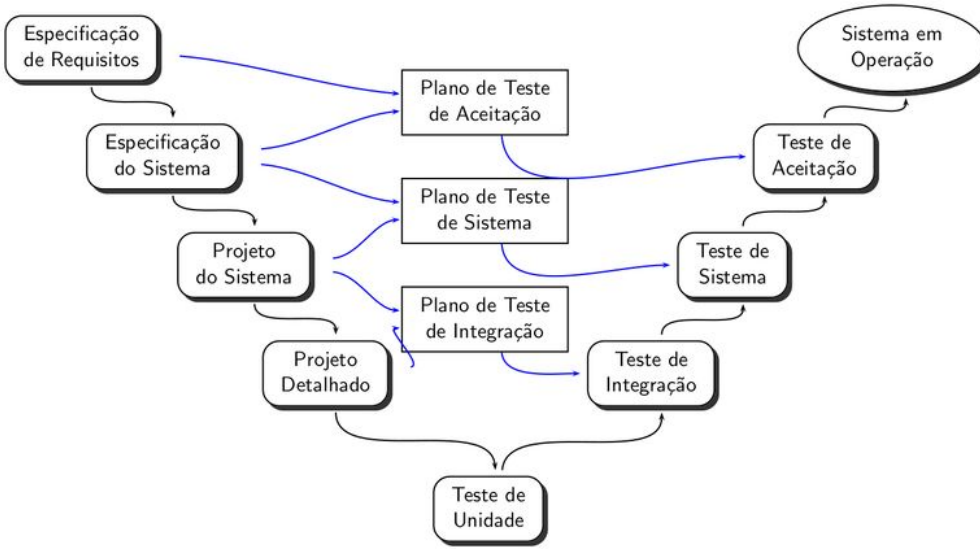
- Diferentes técnicas de teste são apropriadas para diferentes abordagens de engenharia de software, e etapas de projeto
- O teste é feito pelo desenvolvedor do software e por um grupo independente de teste.
- O teste e a depuração são atividades diferentes, mas a depuração deve ser associada com alguma estratégia de teste

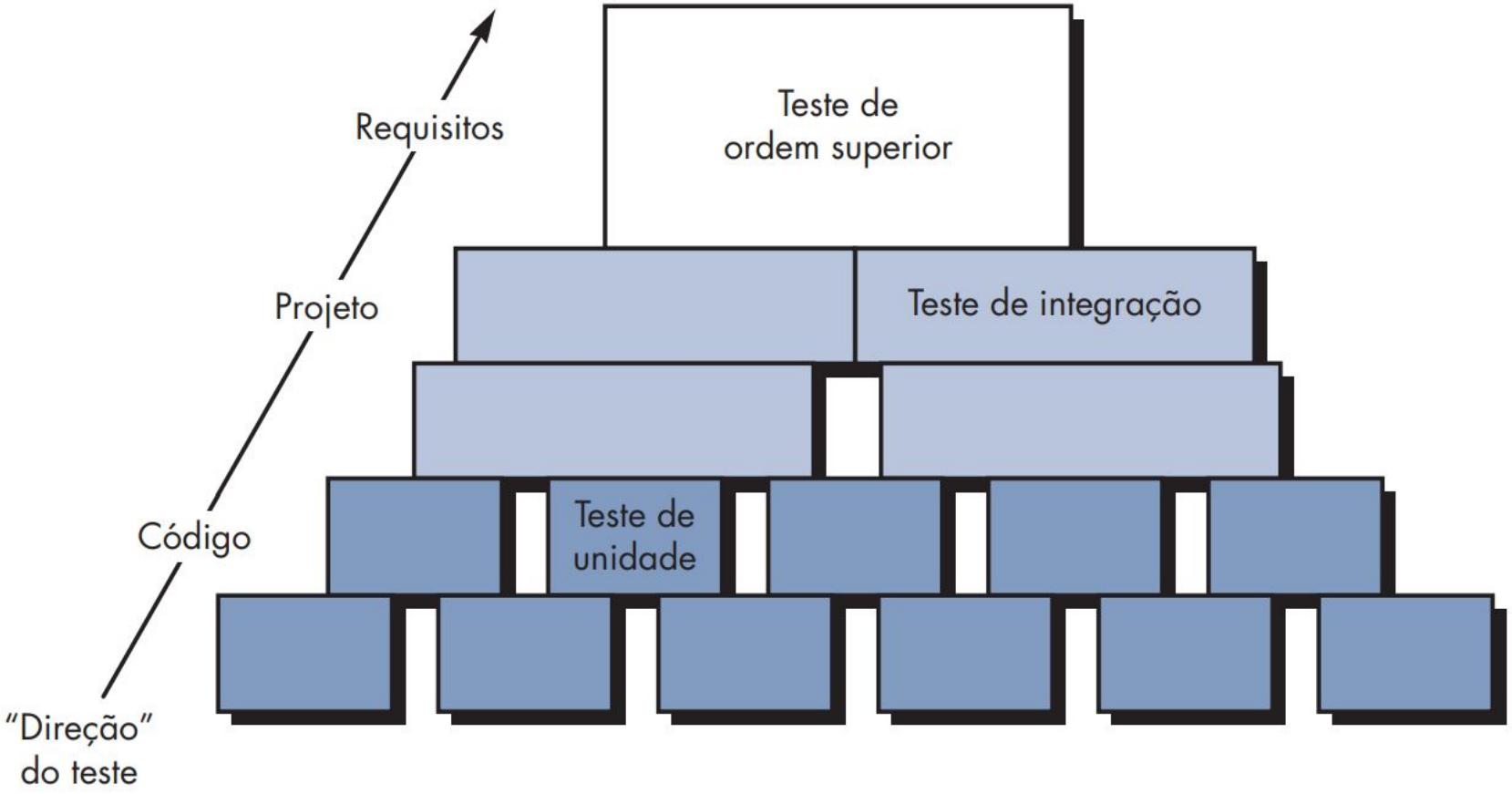
Estratégias de Teste de Software

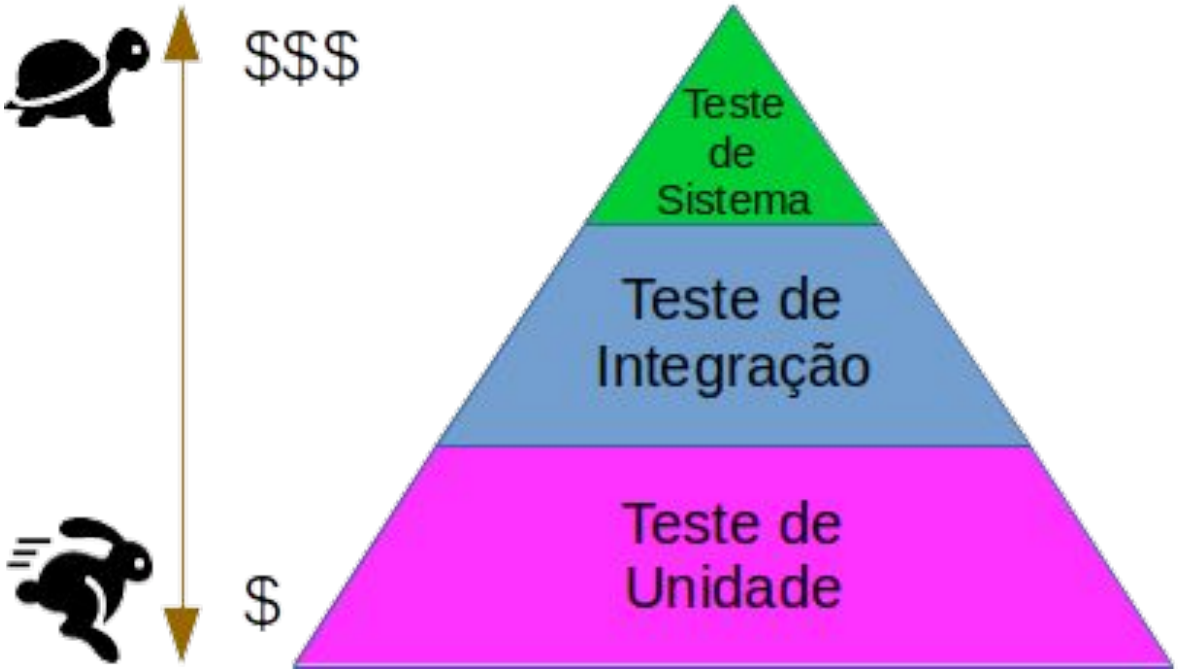
Tom Gilb (Gilb. 1995) argumenta que uma estratégia de teste de software terá sucesso quando os testadores de software:

- *Especificarem os requisitos do produto de uma maneira quantificável, muito antes de começar o teste;*
- *Definirem explicitamente os objetivos do teste.* Os objetivos específicos do teste deverão ser definidos em termos mensuráveis.
- *Criarem software "robusto" que seja projetado para testar-se a si próprio.* Isto é, o software deve ser capaz de diagnosticar certas classes de erros.
- *Usarem revisões técnicas eficazes como filtro antes do teste*


Teste de sistema
Teste de validação
Teste de integração
Teste de unidade
Código
Projeto
Requisitos
Engenharia de sistemas


Especificação de Requisitos
Especificação do Sistema
Projeto do Sistema
Projeto Detalhado
Plano de Teste de Aceitação
Plano de Teste de Sistema
Plano de Teste de Integração
Teste de Unidade
Teste de Integração
Teste de Sistema
Teste de Aceitação
Sistema em Operação


Requisitos
Projeto
Código
"Direção"
do teste
Teste de
ordem superior
Teste de integração
Teste de
unidade



Maior granularidade
Menor quantidade
Mais lentos
Maior custo

Menor granularidade
Maior quantidade
Mais rápidos
Menor custo

Níveis de Teste

- **Teste de Unidade / Unitário**
- **Teste de Componentes**
- **Teste de Integração**
- **Teste de Sistema**
- **Teste de Regressão**
- **Teste de Aceitação**

Técnicas de Teste

- **Teste Caixa-Branca / *White-Box* / Estrutural**
- **Teste Caixa-Preta / *Black-Box* / Funcional**

Tipos de Teste

- **Teste de Usabilidade**
- **Teste de Desempenho / *Performance***
- **Teste de Carga**
- **Teste de Estresse / Esforço**
- **Teste de Segurança**

Sobre Qualidade - Fatores de qualidade ISO 9126

O padrão ISO 9126 foi desenvolvido como uma tentativa de identificar os atributos fundamentais de qualidade para software de computador. O padrão identifica seis atributos fundamentais de qualidade:

- Funcionalidade. O grau com que o software satisfaz às necessidades declaradas conforme indicado pelos seguintes subatributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança.
- Confiabilidade. A quantidade de tempo que o software fica disponível para uso conforme indicado pelos seguintes subatributos: maturidade, tolerância a falhas, facilidade de recuperação.

Fatores de qualidade ISO 9126

- Usabilidade. O grau de facilidade de utilização do software conforme indicado pelos seguintes subatributos: facilidade de compreensão, facilidade de aprendizagem, operabilidade.
- Eficiência. O grau de otimização do uso, pelo software, dos recursos do sistema conforme indicado pelos seguintes subatributos: comportamento em relação ao tempo, comportamento em relação aos recursos.

Fatores de qualidade ISO 9126

- Facilidade de manutenção. A facilidade com a qual uma correção pode ser realizada no software conforme indicado pelos seguintes subatributos: facilidade de análise, facilidade de realização de mudanças, estabilidade, testabilidade.
- Portabilidade. A facilidade com a qual um software pode ser transposto de um ambiente a outro conforme indicado pelos seguintes subatributos: adaptabilidade, facilidade de instalação, conformidade, facilidade de substituição

Dimensões de qualidade de Garvin

Embora as oito dimensões de qualidade de Garvin não tenham sido desenvolvidas especificamente para software, elas podem ser aplicadas quando se considera qualidade de software (Pressman 2011,p.360)

As oito dimensões de Gravin consideram que a qualidade por ser um conceito complexo, resulta de múltiplas dimensões, que se constroem desde a visão do desenvolvedor, passando pela visão de produto e negócio, até a do usuário sobre o tal produto.

Qualidade do Desempenho e dos Recursos

- **Qualidade do desempenho:** O software fornece todo o conteúdo, funções e recursos que são especificados como parte do modelo de requisitos de forma a gerar valor ao usuário final?

- **Qualidade dos recursos:** O software fornece recursos que surpreendem e encantam usuários finais que os utilizam pela primeira vez?

Confiabilidade e Conformidade

O software fornece todos os recursos e capacidades sem falhas?
Está disponível quando necessário?
Fornece funcionalidade sem a ocorrência de erros?

Confiabilidade do Software é a probabilidade que o software não causará uma falha no sistema por um tempo especificado, sob condições determinadas.

O software está de acordo com os padrões de software locais e externos relacionados com a aplicação?
Segue as convenções de projeto e codificação de fato?

Por exemplo, a interface com o usuário está de acordo com as regras de projeto aceitas para seleção de menus ou entrada de dados?

Durabilidade e Manutenibilidade

- Durabilidade. O software pode ser mantido (modificado) ou corrigido (depurado) sem a geração involuntária de efeitos colaterais indesejados?
  As mudanças farão com que a taxa de erros ou a confiabilidade diminuam com o passar do tempo?

- Facilidade de manutenção. O software pode ser mantido (modificado) ou corrigido (depurado) em um período de tempo aceitável e curto? O pessoal de suporte pode obter todas as informações necessárias para realizar alterações ou corrigir defeitos?

Estética e Percepção

- Estética. Não há dúvida nenhuma de que cada um de nós tem uma visão diferente e muito subjetiva do que é estética. Mesmo assim, a maioria de nós concordaria que uma entidade estética tem certa elegância, um fluir único e uma “presença” que são difíceis de quantificar mas evidentes. Um software estético possui essas características.

- Percepção. Em algumas situações, temos alguns preconceitos que influenciarão nossa percepção de qualidade. Se acaso já tivemos outras experiências relacionadas com determinado produto, certamente nossa percepção será influenciada pela experiência passada, seja ela boa ou ruim.


Facilidade de manutenção
Flexibilidade
Testabilidade
Portabilidade
Reusabilidade
Interoperabilidade
REVISÃO DO PRODUTO
TRANSIÇÃO DO PRODUTO
OPERAÇÃO DO PRODUTO
Correção
Confiabilidade
Usabilidade
Integridade
Eficiência

OBRIGADO

daciofmf@gmail.com