TESTES DURANTE O CICLO DE VIDA DO SOFTWARE

Prof. Me. Rober Marcone Rosi

SUMÁRIO

- 2.1 - Modelos de Desenvolvimento de Software
  - Modelo V
  - Modelos iterativos de desenvolvimento
  - Teste dentro de um modelo de ciclo de vida
- 2.2 - Níveis de Teste
  - Teste de Componente
  - Teste de Integração
  - Teste de Sistema
  - Teste de Aceite
- 2.3 - Tipos de Teste: o alvo do teste
  - Teste de Função (Teste funcional)
  - Teste de características do produto de software (testes não funcionais)
  - Teste de estrutura/arquitetura do software (teste estrutural)
  - Teste relacionado a mudanças (teste de confirmação e regressão)
- 2.4 - Teste de Manutenção

DESENVOLVIMENTO DE SOFTWARE

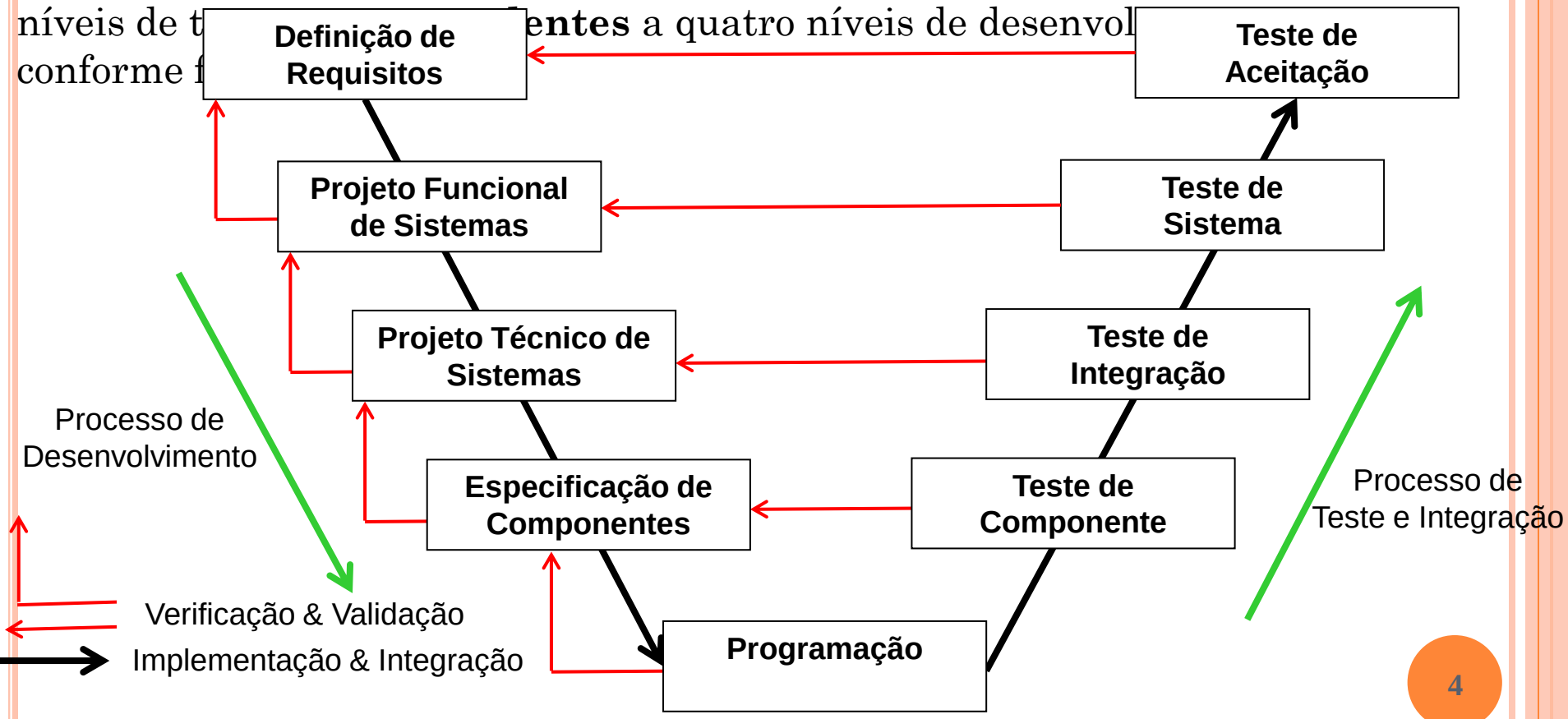
- Não existe teste isolado; a atividade de teste está intimamente relacionada com as atividades de desenvolvimento do software.

- Modelos de ciclo de vida de desenvolvimento diferentes necessitam de abordagens diferentes para testar.

O MODELO V

- Apesar das variações do Modelo V, um tipo comum deste modelo usa quatro

níveis de t**entes** a quatro níveis de desenvol
conforme f



O MODELO V

- Na prática, um Modelo V, pode ter mais, menos ou diferentes níveis de desenvolvimento e teste, dependendo do projeto e do produto. Por exemplo, pode haver teste de integração de componentes após o teste de um componente, e teste de integração de sistemas após o teste de sistemas.

- Produtos de trabalho de software (como cenário de negócios ou casos de uso, especificação de requisitos, documentos de modelagem e código) produzidos durante o desenvolvimento muitas vezes são a base do teste em um ou mais nível de teste.

MODELOS ITERATIVOS DE DESENVOLVIMENTO

- Desenvolvimento iterativo é o processo que estabelece os requisitos, modelagem, construção e teste de um sistema, realizada como uma série de desenvolvimentos menores.
- Exemplos de desenvolvimento iterativo:
  - Prototipagem
  - Desenvolvimento rápido de aplicação (RAD)
  - Rational Unified Process (RUP)
  - Modelos ágeis de desenvolvimento
- O produto resultante de uma iteração pode ser testado em vários níveis como parte do seu desenvolvimento.
- Um incremento, adicionado a outros desenvolvidos previamente, forma um sistema parcial em crescimento, que também deve ser testado.
- Teste de regressão tem sua importância aumentada a cada iteração. Verificação e validação podem ser efetuadas a cada incremento.

TESTE DENTRO DE UM MODELO DE CICLO DE VIDA

- Os itens abaixo indicam as características para um bom teste para qualquer modelo de ciclo de vida:
  - **Para todas as atividades do desenvolvimento há uma atividade de teste correspondente.**
  - Cada nível de teste tem um objetivo específico daquele nível.
  - A análise e modelagem do teste para um dado nível de teste devem começar durante a atividade de desenvolvimento correspondente.
  - Testadores devem se envolver na revisão de documentos o mais cedo possível utilizando as primeiras versões disponíveis ao longo ciclo de desenvolvimento.

NÍVEIS DE TESTE

- Os níveis de teste descritos no *syllabus* são:
  - **Teste de Componente (ou Unitário)**
  - **Teste de Integração**
  - **Teste de Sistema**
  - **Teste de Aceite (ou Aceitação)**
- Os níveis de teste também podem ser chamados de **estratégias de teste** ou **estágios do processo de teste** (PRESSMAN, 2006; SOMMERVILLE, 2004) .

- Para cada nível de teste, os seguintes aspectos podem ser identificados:
  - seus objetivos genéricos
  - os produtos de trabalho utilizados como referência para derivar os casos de testes (ex.: base do teste)
  - o objeto do teste (o que está sendo testado)
  - defeitos e falhas típicas a se encontrar
  - testes (“*harness”)* e ferramentas de suporte e abordagens e responsabilidades específicas.

TESTE HARNESS

- No teste de software, um teste Harness ou estrutura de teste automatizado é uma coleção de software e dados de teste configurados para testar uma unidade de programa executando-a em condições variadas e monitorando seu comportamento e resultados. Ele tem duas partes principais: o mecanismo de execução de teste e o repositório de script de teste.
- Os testes Harness permitem a automação dos testes. Eles podem chamar funções com parâmetros fornecidos e imprimir e comparar os resultados com o valor desejado. O teste Harness é um gancho para o código desenvolvido, que pode ser testado usando uma estrutura de automação.
- Um teste Harness deve permitir a execução de testes específicos (isso ajuda na otimização), orquestrar um ambiente de tempo de execução e fornecer uma capacidade de análise de resultados.

TESTE HARNESS

- Os objetivos típicos de um teste Harness são:
  - Automatizar o processo de teste.
  - Executar suites de casos de teste.
  - Gerar relatórios de teste associados.

- Um teste Harness pode fornecer alguns dos seguintes benefícios:
  - Aumento de produtividade devido à automação do processo de teste.
  - Maior probabilidade de ocorrer o teste de regressão.
  - Maior qualidade de componentes de software e aplicativos.
  - Certifique-se de que as execuções de teste subsequentes sejam duplicatas exatas das anteriores.
  - O teste pode ocorrer em horários em que o escritório não tem pessoal (por exemplo, à noite)
  - Um script de teste pode incluir condições e / ou usos que são de outra forma difíceis de simular (carregar, por exemplo)

TESTE HARNESS

- Uma definição alternativa de um teste Harness é um software construído para facilitar o teste de integração.
- Onde os stubs de teste são normalmente componentes do aplicativo em desenvolvimento e são substituídos por um componente funcional conforme o aplicativo é desenvolvido (design de cima para baixo), os testes Harness são externos ao aplicativo que está sendo testado e simulam serviços ou funcionalidades não disponíveis em um ambiente de teste.
- Por exemplo, se você estiver construindo um aplicativo que precisa fazer interface com um aplicativo em um computador mainframe, mas nenhum está disponível durante o desenvolvimento, um equipamento de teste pode ser construído para ser usado como um substituto.
- Um equipamento de teste pode ser parte de uma entrega de projeto. Ele é mantido fora do código-fonte do aplicativo e pode ser reutilizado em vários projetos.
- Como um equipamento de teste simula a funcionalidade do aplicativo - ele não tem conhecimento de suites de teste, casos de teste ou relatórios de teste.
- Essas coisas são fornecidas por uma estrutura de teste e ferramentas de teste automatizadas associadas.

TESTE DE COMPONENTE

- Teste de componentes procura defeitos e verifica o funcionamento do software (ex.: módulos, programas, objetos, classes, etc.) **que são testáveis separadamente**.

- **Pode ser feito isolado do resto do sistema**, dependendo do contexto do ciclo de desenvolvimento e do sistema. Controladores (*"drivers") e simuladores ("stubs") podem ser usados*.

- **Tipicamente, teste de componente ocorre com acesso ao código que está sendo testado e no ambiente de desenvolvimento**, assim como um framework de teste de unidade ou ferramenta de depuração *"debugging"*.

- **Na prática, envolve o programador do código.**

- **Defeitos são normalmente corrigidos assim que são encontrados sem registrar formalmente os incidentes.**

TESTE DE COMPONENTE

- Teste de componente pode incluir teste de funcionalidade e características específicas não funcionais tais como comportamento dos recursos (ex.: falta de memória) e testes de robustez, além de teste estrutural (cobertura de código).

- Casos de teste são derivados dos produtos de trabalho como, por exemplo, especificação de componente, modelagem do software ou modelo de dados.

- Uma abordagem no teste de componente consiste em preparar e automatizar os casos de testes antes de codificar. Isto é chamado de abordagem de teste antecipado ou desenvolvimento dirigido a teste.

- Esta abordagem é essencialmente iterativa e é baseada em ciclos de elaboração de casos de testes. À medida que são construídas e integradas pequenas partes do código, são executados testes de componente até que eles passem.

TESTE DE INTEGRAÇÃO

- **É caracterizado por testar as interfaces entre os componentes, interações de diferentes partes de um sistema, como o sistema operacional, arquivos, hardware ou interfaces entre os sistemas.**

- Pode haver mais que um nível de teste de integração, que pode ser utilizado em objetos de teste de tamanho variado. Por exemplo:
  - Teste de integração de componente testa interações entre componentes de software e é realizado após o teste de componente.
  - Teste de integração de sistemas testa interação entre diferentes sistemas e pode ser realizado após o teste de sistema. Neste caso a área de desenvolvimento pode controlar apenas um lado da interface, de forma que mudanças podem causar instabilidades. Processos de negócios implementados como fluxogramas podem envolver uma série de sistemas. Problemas relacionados a múltiplas plataformas podem ser significativos.

- Quanto maior o escopo da integração, maior a dificuldade de isolar as falhas para componentes ou sistemas específicos, fato que pode representar um aumento no risco.

TESTE DE INTEGRAÇÃO

- Estratégias sistemáticas de integração podem ser baseadas na

arquitetura do sistema (*top-down e bottom-up), funções, sequências de processamento de transações, entre outros aspectos do sistema ou componente.* ***Visando reduzir o risco de encontrar defeitos tardiamente, a integração deve, preferencialmente, ser incremental e não "big bang".***

- Teste de características não funcionais específicas (por exemplo, performance) pode ser incluído nos testes de integração.

- A cada estágio da integração, os testadores concentram somente na integração propriamente. Por exemplo, o módulo A está sendo integrado com o módulo B e, neste caso, o foco é a comunicação entre os módulos e não suas funcionalidades. Tanto testes funcionais quanto estruturais podem ser utilizados.

- Idealmente, os testadores devem compreender a arquitetura e influenciar no planejamento da integração. Se o teste de integração for planejado antes que os componentes ou sistemas estejam prontos, eles podem ser preparados visando um teste mais eficiente.

TESTE DE SISTEMA

- **Se refere ao comportamento de todo do sistema / produto definido pelo escopo de um projeto ou programa de desenvolvimento.**

- No teste de sistema, **o ambiente de teste deve corresponder o máximo possível ao objetivo final, ou o ambiente de produção**, para minimizar que os riscos de falhas específicas de ambiente não serem encontradas durante o teste.

- Testes de sistemas podem ser baseados em especificação de riscos e/ou de requisitos, processos de negócios, casos de uso, dentre outras descrições de alto nível do comportamento, interações e recursos do sistema.

TESTE DE SISTEMA

- **Teste de sistema deve tratar requisitos funcionais e não funcionais do sistema.**
- Os requisitos podem estar como um texto ou diagramas.
- Testadores devem também lidar com requisitos incompletos ou não documentados.
- **Teste de sistema em requisitos funcionais deve inicialmente utilizar a técnica baseada em especificação mais apropriada (caixa-preta)** de acordo com a característica do sistema a ser testado. Por exemplo, uma tabela de decisão pode ser criada por combinações de efeitos descritos em regras de negócio.
- **A seguir, técnica baseada na estrutura (caixa-branca) pode ser utilizada** para avaliar a eficácia do teste com respeito ao elemento estrutural, assim como estrutura do menu ou página web.
- Uma equipe de teste independente é frequentemente responsável pelo teste de sistema.

TESTE DE ACEITE

- **Teste de aceite frequentemente é de responsabilidade do cliente ou do usuário do sistema; os interessados (*stakeholders)* também podem ser envolvidos. O objetivo do teste de aceite é estabelecer a confiança no sistema, parte do sistema ou uma característica não específica do sistema.**

- Procurar defeitos não é o principal foco do teste de aceite. Teste de aceite pode avaliar a disponibilidade do sistema para entrar em produção, apesar de não ser necessariamente o último nível de teste. Por exemplo, teste de integração em larga escala pode vir após o teste de aceite de um sistema.

- O teste de aceite pode ser realizado em mais de um único nível de teste. Por exemplo:
  - Um pacote (COTS - *Commercial Off the Shelf*) de software ter um teste de aceite quando é instalado ou integrado.
  - Teste de aceite de usabilidade de um componente pode ser feito durante o teste de componente.
  - Teste de aceite de uma nova funcionalidade pode vir antes do teste de sistema.

FORMAS DE TESTE DE ACEITE

- As formas de teste de aceite incluem tipicamente os seguintes:
  - Teste de Aceite de Usuário
  - Teste Operacional de Aceite
  - Teste de aceite de contrato e regulamento
  - Teste Alfa e Beta (ou teste no campo)

FORMAS DE TESTE DE ACEITE

- **Teste de Aceite de Usuário**
  - Normalmente verifica se o sistema está apropriado para o uso por um usuário com perfil de negócio.

- **Teste Operacional de Aceite**
  - O aceite do sistema pelo administrador dos sistemas inclui:
    - Teste de Backup/Restore.
    - Recuperação de Desastre.
    - Gerenciamento de Usuário.
    - Tarefas de manutenção.
    - Checagens periódicas de vulnerabilidades de segurança.

FORMAS DE TESTE DE ACEITE

- **Teste de aceite de contrato e regulamento**
  - é realizado verificando-se algum critério de aceite incluso em contrato na produção de software encomendado.
  - O critério de aceite deve ser definido quando o contrato é assinado.
  - Teste de aceite de regulamento é quando se verifica a necessidade de adesão a algum regulamento de acordo com outras normas (ex.: segurança, governamental, legislação).

FORMAS DE TESTE DE ACEITE

- **Testes Alfa e Beta (ou teste no campo)**
  - Desenvolvedores de softwares comerciais ou pacotes, muitas vezes precisam obter um feedback de clientes em potencial existente no mercado antes que o software seja colocado à venda comercialmente.
  - **Teste Alfa** é feito no "site" da organização em que o produto foi desenvolvido, ou seja, **é conduzido na instalação do desenvolvedor com os usuários finais**, com o desenvolvedor observando o usuário final.
  - **Teste Beta**, ou teste no campo, é feito pelas pessoas em suas próprias localidades, ou seja, **é conduzido nas instalações do cliente, e, diferente do teste alfa, o desenvolvedor não está presente e o ambiente não pode ser controlado pelo desenvolvedor.**
  - **Ambos os testes são feitos pelos clientes em potencial e não pelos desenvolvedores do produto**.

- Organizações podem utilizar outros termos como Teste de Aceite de Fábrica e Teste de Aceite no "site", para sistemas que são testados **antes e após** terem sido movidos ao "site" do cliente.

TIPOS DE TESTE: O ALVO DO TESTE

- Um grupo de atividades de teste pode ser direcionado para verificar o sistema (ou uma parte do sistema) com base em um motivo ou alvo específico.

- Cada tipo de teste tem **foco em um objetivo particular**, que pode ser:
  - o teste de uma funcionalidade, a ser realizada pelo software;
  - uma característica da qualidade não funcional, tal como a confiabilidade ou usabilidade,
  - a estrutura ou arquitetura do software ou sistema;
  - mudanças relacionadas, ex.: confirmar que os defeitos foram solucionados (teste de confirmação) e procurar por mudanças inesperadas (teste de regressão).

- Modelos do software podem ser elaborados e/ou usados no teste estrutural ou funcional. Por exemplo, para o teste funcional, um diagrama de fluxo de processo, um diagrama de transição de estados ou uma especificação do programa, e para teste estrutural um diagrama de controle de fluxo ou modelo de estrutura do menu.

TIPOS DE TESTE: O ALVO DO TESTE

- Os tipos de teste podem ser:
  - Teste de **Função** (Teste funcional)
  - Teste de **características** do produto de software (testes não funcionais)
  - Teste de **estrutura/arquitetura** do software (teste estrutural)
  - Teste relacionado a **mudanças** (teste de confirmação e regressão)

TIPOS DE TESTE: O ALVO DO TESTE

Teste de Função (Teste funcional)

- As funções que um sistema, subsistema ou componente devem realizar podem ser descritas nos seguintes produtos de trabalho: especificação de requisitos; casos de uso, especificação funcional, ou podem não estar documentados. As funções representam "o que" o sistema faz.
- Testes funcionais são baseados em funções (descritas nos documentos ou compreendidas pelos testadores), e devem ser realizados em todos os níveis de teste (ex.: teste de componente deve ser baseado na especificação do componente).
- Técnicas baseadas em especificação podem ser utilizadas para derivar as condições de teste e casos de testes a partir da funcionalidade do software ou sistema. Teste funcional considera o comportamento externo do software (teste caixa-preta).
- Um tipo de teste funcional chamado teste de segurança, investiga as funções (ex.: um *"firewall")* relacionados à detecção de ameaça de vírus ou de ações mal intencionadas.

TIPOS DE TESTE: O ALVO DO TESTE

Teste de características do produto de software (testes não funcionais)

- Testes não funcionais incluem, mas não se limitam a: teste de performance; teste de carga; teste de estresse; teste de usabilidade; teste de interoperabilidade; teste de manutenibilidade; teste de confiabilidade e teste de portabilidade. É o teste de "como" o sistema trabalha.
- Testes não funcionais podem ser realizados em todos os níveis de teste. O termo teste não funcional descreve que o teste é executado para medir as características que podem ser quantificadas em uma escala variável, como o tempo de resposta em um teste de performance.
- Estes testes podem ser referenciados a um modelo de qualidade como definido na norma *"Engenharia de Software – Qualidade de Produto de Software" (ISO 9126).*

TIPOS DE TESTE: O ALVO DO TESTE

Teste de estrutura/arquitetura do software (teste estrutural)

- Teste estrutural (caixa-branca) pode ser feito em todos os níveis de testes. Recomenda-se utilizar as técnicas estruturais após as técnicas baseadas em especificação, já que ela auxilia a medição da eficiência do teste através da avaliação da cobertura de um tipo de estrutura.
- Cobertura é a extensão que uma estrutura foi exercitada por um conjunto de testes, expresso como uma porcentagem de itens cobertos. Se a cobertura não atinge 100%, então mais testes devem ser construídos a fim de testar aqueles itens que não foram contemplados para, desta forma, aumentar a cobertura.
- Em todos os níveis de teste, mas especialmente no teste de componente e teste de integração de componentes, **ferramentas podem ser usadas** para medir a cobertura do código dos elementos assim como as declarações ou decisões. Teste estrutural deve ser baseado na arquitetura do sistema, como uma hierarquia de chamadas.
- Teste de estrutura também pode ser aplicado no sistema, integração de sistema ou nível de teste de aceite (por exemplo, para modelos de negócios ou estrutura de menu).

TIPOS DE TESTE: O ALVO DO TESTE

Teste relacionado a mudanças (teste de confirmação e regressão)

- **Teste de Confirmação:**
  - quando um defeito é detectado e resolvido, o software pode ser retestado para confirmar que o defeito original foi realmente removido. Isto é chamado de teste de confirmação. Depurar (resolver defeitos) é uma atividade do desenvolvimento, e não uma atividade do teste.

- **Teste de Regressão:**
  - é o teste repetido de um programa que já foi testado, após sua modificação, para descobrir a existência de algum defeito introduzido ou não coberto originalmente como resultado da mudança.
  - Estes defeitos podem estar no software ou em um componente, relacionado ou não ao software. É realizado quando o software, ou seu ambiente é modificado.
  - A quantidade de teste de regressão é baseada no risco de não se encontrar defeitos no software que estava funcionando previamente.

TIPOS DE TESTE: O ALVO DO TESTE

Teste relacionado a mudanças (teste de confirmação e regressão)

- Os testes devem ser repetíveis se forem utilizados nos testes de confirmação e para suportar o teste de regressão.
- Teste de regressão pode ser realizado em todos os níveis de teste, e se aplicam aos testes funcionais, não funcionais e estruturais.
- Testes de regressão são executados muitas vezes e geralmente desenvolve-se vagarosamente, o que faz com que seja um forte candidato à automação.

TESTE DE MANUTENÇÃO

- Uma vez desenvolvido, um sistema pode ficar ativo por anos ou até mesmo décadas.
- Durante este tempo o sistema e seu ambiente podem ser corrigidos, modificados ou completados.
- **Teste de manutenção é realizado no mesmo sistema operacional e é iniciado por modificações, migrações ou retirada de software ou sistema.**

- Alguns exemplos de modificações incluem
  - melhorias planejadas (ex.: baseadas em *"releases"),*
  - mudanças corretivas e emergenciais
  - de mudanças de ambiente, como atualização em sistema operacional ou banco de dados, e
  - correções ("patches") para expor e encontrar vulnerabilidades do sistema operacional.

TESTE DE MANUTENÇÃO

- **Teste de manutenção por migração** (ex.: de uma plataforma a outra) pode incluir testes operacionais do novo ambiente tanto quanto a mudança de software.
- **Teste de manutenção para retirada de um sistema** pode incluir o teste de migração de dados, ou arquivamento se longos períodos de retenção de dados forem necessários.
- **Além de testar o que foi alterado, o teste de manutenção inclui teste de regressão massivo para as partes do sistema que não foram testadas.**
- O escopo do teste de manutenção está relacionado ao risco da mudança, o tamanho do sistema existente e o tamanho da mudança.
- Dependendo da mudança, o teste de manutenção pode ser feito em todos ou alguns níveis, e em todos ou alguns tipos de testes.
- A determinação de como um sistema pode ser afetado por mudanças é chamado de análise de impacto, e pode ser usado para ajudar a decidir quantos testes de regressão serão realizados.
- Teste de manutenção pode se tornar uma tarefa complicada se as especificações estiverem desatualizadas ou incompletas.

Referências

- Baseado no material da Profa. Denise Togneri e do

Prof. Ralf Luis de Moura