Professor:	JUCEMAR FORMIGONI CÂNDIDO					**e-mail**:	238879@profe.sed.sc.gov.br		
Curso/Matriz:	3094 – EMIEP / Téc. em Informática								
Disciplina: (Componente Curricular)	Teste de Software								
Ano/Série/Mód.:	3º	**Turma(s):**	5?	**Turno:**	Diurno	**Ano Letivo**:		**Modalidade**:	Ensino Médio

Ferramentas para Testes de Software

O processo de realização dos testes é otimizado com softwares voltados para otimizar a gestão dos recursos da empresa, detectar e corrigir problemas de performance e de integração com outros sistemas da empresa. Para escolher uma ferramenta vai depender do seu foco.

Então é necessário se questionar:

- ✓ Você precisa de uma ferramenta para fazer testes unitários?
- ✓ Você precisa de uma ferramenta para suporte a BDD?
- ✓ Você precisa de uma ferramenta para automatizar testes de interface?
- ✓ Você precisa de uma ferramenta para qual tipo de teste?
- ✓ Você precisa de uma ferramenta para gerenciamento de testes?
- ✓ Você precisa de uma ferramenta para teste de regressão?

Apesar de não existir uma categorização amplamente difundida das ferramentas de teste, a experiência tem mostrado que elas são normalmente agrupadas em 8 áreas distintas:

- ✓ Ferramentas de automação de testes de regressão;
- ✓ Ferramentas para gestão de defeitos;
- ✓ Ferramentas para testes de Performance/Stress;
- ✓ Ferramentas manuais;
- ✓ Ferramentas de rastreabilidade;
- ✓ Ferramentas de cobertura de código;
- ✓ Ferramentas para gestão de testes;
- ✓ Ferramentas de apoio à execução dos testes.

***Obs.:** Então não é uma tarefa fácil escolher qual ferramenta de automação de teste escolher, pois cada uma possui funcionalidades que cada equipe vai precisar. Por ser verdade, listamos abaixo algumas ferramentas para facilitar sua decisão na hora de escolher uma ajuda para sua automação de testes. Lembrando que o desenvolvimento, inspeção e o teste de unidade são as três partes do teste de códigos.*

Ferramentas de teste de software: Quando e por que automatizar?

No desenvolvimento de projetos é muito importante ter ferramentas de teste para fazer com que tudo esteja dentro das expectativas. A realização de testes, durante o andamento, fará com que o sistema funcione adequadamente quando estiver implantado e funcionando conforme planejado na empresa. Eles, vão garantir que o software passe por ajustes e manutenções enquanto ainda é possível detectar falhas. Caso os erros apareçam quando o software já estiver rodando, grande parte do trabalho terá de ser refeito, atrasando a implantação do novo sistema, trazendo prejuízos pela suspensão de processos. Além disso, existem ferramentas de teste de software e técnicas de automatização que podem ser utilizadas para facilitar estas etapas de verificação.

Quando fazer a automação de testes?

Os testes de software são realizados de acordo com as especificações dos "***requisitos de software***", documento essencial que estima custos do projeto, prazos de desenvolvimento, modelagem e prototipagem. Esses requisitos são fundamentais para garantir o desempenho do sistema. Além disso, é por meio deles que a equipe de desenvolvimento vai perceber a necessidade de manutenções no sistema e analisar o impacto das mudanças necessárias.

Sendo assim, a realização dos testes de desempenho ocorre durante o desenvolvimento do software, para que o tempo seja otimizado com a correção de bugs por etapas.

Ainda, evita prejuízos financeiros e perda de credibilidade, caso o software tenha um mau funcionamento depois de pronto.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Por que utilizar ferramentas de teste de software?

Automatizar os testes de software é uma atividade que merece atenção, pois a qualidade do produto final depende disto. O processo é repetitivo, pois é necessário fazer ajustes, testar, reajustar, … Contudo, é possível fazer isso de uma forma inovadora. Recomenda-se, pois além de otimizar o trabalho da equipe de desenvolvedores, auxilia outros quesitos:

- ✓ Conferem segurança e confiabilidade ao produto;
- ✓ Otimizam a gestão dos recursos ao longo do projeto;
- ✓ Evitam o excesso de trabalho manual;
- ✓ Identificam problemas de ordens diversas;
- ✓ Oferecem tempo hábil para fazer reparos;
- ✓ Garantem feedback em todo o processo de desenvolvimento;
- ✓ Podem render um grande volume de dados;
- ✓ Ajudam a checar resultados e configurações;
- ✓ Garantem uma rede de segurança ao sistema;
- ✓ Ajudam a melhorar o ROI.

Outro motivo para automatizar os testes de software é que existem diversos **tipos de testes** para serem realizados e se forem de forma manual tornam-se muito complexo e moroso.

Tipos de testes de software e por que automatizá-los?

É primordial corresponder à qualidade em seu projeto. Um projeto de sucesso se torna um grande desafio. Por isso, a necessidade de utilizar diferentes tipos de testes durante as etapas de desenvolvimento.

Um outro aspecto importante é a grande demanda de softwares, que vem sempre junto com prazos de entrega a tempo. Com isto, pode-se correr o risco de um projeto instável, e até deixar erros simples desapercebido (ex.: Data de nascimento futura), ou riscos e problemas mais à frente. Por isso, a importância de entender cada um dos testes e a forma como aplicar.

O que é um teste de software?

Os testes fazem parte do processo de desenvolvimento do projeto/programa, podendo ser feito pelos desenvolvedores ou, em por profissionais especializados na área. O procedimento tem objetivo antecipar e corrigir falhas e bugs que apareceriam para o usuário final. Apesar de processos simples, mas são essenciais, e evita o famoso “apagar incêndios”. Imagine o cliente lidar com instabilidades ou dificuldades em acessar o layout defeituoso. Para evitar surpresas desagradáveis, recorra à diferentes tipos de testes para garantir a certificação do sistema e verificar se o funcionamento está conforme o planejado. Tipos de Testes de Softwares existentes:

Testes de caixa branca

O profissional por conhecer o código fonte, observa com atenção as etapas do código, analisando o caminho do fluxo dos dados, verificando a passagem correta nas condições esperadas.

Teste de caixa-preta

Oposto ao anterior, nesse teste não se tem acesso ao código fonte, nem a sua estrutura. É chamado de teste funcional por ser baseado nos requisitos funcionais. Ao testar o código, ficamos atentos à maneira que os usuários acessam a aplicação. É primordial testar todas as combinações possíveis na área de entrada de dados, para que o resultado sai como esperado pelo usuário.

Ex.: *Imagine um portal do convênio médico onde o cliente deve utilizar o seu RG e data de nascimento para o acesso. O programa espera que ambos os campos sejam preenchidos, caso o usuário consiga acesso apenas por um dos campos, significa que existe uma falha na consistência de dados de entrada.*

Este teste pode para casos como:

- ✓ Consistir a entrada de datas futuras em datas de nascimento;
- ✓ Consistir entrada de valores negativos em campos de pagamentos;
- ✓ Verificar o funcionamento dos botões para prosseguir o fluxo de processamento.

Testes de regressão

No desenvolvimento podemos ter as seguintes situações: A inclusão de uma nova funcionalidade pelo cliente ou encontrar um erro na lógica do código fonte. Independentemente, o desenvolvedor fará a alteração na programação. O problema é que, em alguns casos, uma simples mudança compromete toda a lógica já escrita, invalidando quaisquer testes básicos feitos no processo de produção. Para evitar imprevistos, é recomendado que você faça, mesmo que a modificação seja pequena, o teste de regressão.

Teste de usabilidade

Este teste objetiva verificar a experiência do usuário como ponto de vista de consumidor, checando a organização dos itens disponíveis na tela, observar se o layout está correto e se os botões se comunicam corretamente entre as diferentes páginas do sistema. Também, verifica a performance do programa ao executar uma determinada ação. Afinal, é comum nos depararmos com aplicativos que demoram muito tempo para ser carregado. Geralmente o ponto de vista é: “deve ter entrado em Loop”, ou ter executado alguma função inesperada.

Um outro ponto é a análise em plataformas diferentes, diferentes navegadores, dispositivos de tamanhos diferentes, dando uma visão se o layout é responsivo ou não.

Segurança

Os testes de software evoluíram muito com o tempo e um know-how maior foi adquirido pelos profissionais da área. Dentre os novos recursos utilizados para garantir o pleno funcionamento de um programa de computação estão os testes de segurança para verificar a segurança do software no que diz respeito à proteção de ataques diversos, ligado a hackers, vírus aos dados inseridos no software.

Integração

Nesse teste, não se testa as funcionalidades do software, mas a integração entre as diferentes unidades que formam o sistema. Verifica-se aspectos como: Interface e a dependência entre os componentes.

Performance

Testa o desempenho do software. Se os comandos dados respondem rapidamente, se os componentes não demoram muito a carregar e se a experiência do usuário é satisfatória no produto testado. É de suma importância, pois por mais bem programado que um software possa ser, sua performance é que determinará o uso satisfatório das funções planejadas.

Instalação

O teste de instalação verifica se sob diferentes condições como pouco espaço de memória, interrupções no sistema e demais entraves que podem comprometer esse processo, o programa consegue ser instalado ou se cede facilmente a essas limitações. Você já deve ter vivido a experiência em celular quando resolve instalar um App e o aparelho trava ou a instalação é interrompida por alguns dos fatores citados acima.

Manutenção

Um software não dever ser feito para durar pouco tempo, tampouco uma única versão. Atualizações constantes são necessárias a fim de aprimorar os recursos do programa e a própria experiência do usuário. Esse teste deve averiguar se esses aprimoramentos acontecem com sucesso e se são aceitos pelo sistema. Sem isso, corre-se o risco do software tornar-se defasado e até mesmo inoperante pela falta de atualização.

Funcional

Abrange os testes de caixa branca e caixa-preta. Consiste na capacidade de determinar se o que o software foi programado para fazer está de fato fazendo. Pode ser de forma manual, automática ou um misto dos dois. Várias funções são acessadas e testadas de formas diferentes para encontrar falhas ou mesmo possíveis aprimoramentos no que já está sendo executado. Como o teste de performance, o funcional dá uma noção bastante real de como o software se comportará quando chegar ao usuário, permitindo ajustes importantes antes de atender o seu público final.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Benefícios em automatizar os testes:

Por ser um processo repetitivo e que demanda atenção, é recomendado que a empresa busque a automatização para entregar um trabalho de melhor qualidade.

Evitar o trabalho repetitivo: Normalmente, os testes devem ser executados inúmeras vezes. Ao contar com um funcionário para executar o mesmo procedimento exaustivamente, ele, em algum momento, pode se distrair e influenciar de forma negativa o resultado.

Ter um feedback mais rápido: Quando a empresa precisa de uma resposta rápida, seja para progredir no projeto, seja para fazer novos testes, a automação permite uma entrega contínua, sendo bastante eficiente. Esse benefício é gerado principalmente quando há um planejamento apertado a ser cumprido.

Melhorar a performance de testes de regressão: Em alguns programas esse tipo de teste pode ser frequente, principalmente quando é necessário encontrar um bug escondido no desenvolvimento. O problema é que, mesmo corrigindo a falha encontrada, entre uma alteração e outra, o sistema pode ter uma queda de performance. Nesse caso, a automação torna o processo mais prático para que os desenvolvedores possam analisar, com mais tranquilidade, qual modificação gerou o novo problema.

Economizar tempo: A execução exige a entrada repetida de dados todas as vezes em que ele é executado. Além de evitar a entrada incorreta de dados, o desenvolvedor não precisará se preocupar em colocar a mesma base, já que o processo é feito de forma automática. Por ser uma etapa trabalhosa, algumas empresas acreditam que os testes não são necessários em seu desenvolvimento. O problema é que o lançamento da aplicação precoce pode trazer desde os problemas mais simples, como a instabilidade em seu uso, até os mais graves, como o desaparecimento de dados ou a exposição de informações pessoais. Quando o problema estiver relacionado à segurança, há chances de ele trazer grandes prejuízos financeiros com a manutenção de código.

Elaborando teste de software impecável:

Para elaborar teste realmente eficaz e diferenciado, você deve utilizar outras abordagens além dos que foram elencados. Abaixo, alguns recursos extras que podem ser utilizados:

Teste de Aceitação pelo Usuário

Mesmos após ter feito muitos testes, que se acha suficiente, a precaução nunca é demais para se ter sucesso na sua empreitada, faça então um teste de aceitação do usuário, usando versões beta, para que muitos usuários testem a fim de realizar últimos ajustes antes do software chegar ao mercado.

Teste de Volume

A verificação do volume de dados que o software suporta, deve ser feita sempre após os principais testes. Conte com a ajuda de algumas ferramentas que ajudam a detectar essas limitações como: O BugZilla, Apache JMater e Push Test Maker.

Teste de Stress

O teste de stress busca rotas imprevisíveis no uso do programa, é considerado como uma prova de fogo, pois se o usuário fizer uma rota fora do previsto pela programação no uso do software, como ele reagirá? Foi considerada essa possibilidade no desenvolvimento do software?

Os testes de stress são fundamentais em aplicações em que a eficiência seja uma característica importante. Por exemplo:

- Servidores de arquivos e servidores web, que devem atender a solicitações de um grande número de clientes;
- Aplicações industriais, tais como o controle de uma refinaria de petróleo;
- Jogos de computador, que precisam de um desempenho aceitável para serem viáveis comercialmente.

CEDUP Abílio Paulo – Criciúma – SC – *Tel.: (48)3438-0548/3403-1608/3403-1609*

Disciplina: **Teste de Software**

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Ferramentas de automação de testes mais usadas

Fonte: 10 ferramentas de automação de testes mais usadas - Prime Control – Acesso 03/11/2021).

Numa era onde tudo é automatizado, testadores de software tem demandado cada vez mais ferramentas de automação de testes. Escolha uma delas para sua necessidade:

Selenium: É uma ferramenta muito versátil e muito usada do mercado. Ferramenta open source usada principalmente para aplicações Web. A estrutura de testes do Selenium atua em plataformas e Browsers como: Linux, Mac, Windows, Firefox, Chrome, Internet Explorer, assim como em HeadLess Browsers. O Selenium IDE, um add-on de navegador, permite gravar e reproduzir, o que significa que o testador pode usar estas funcionalidades até mesmo se estiver aprendendo a usar o Selenium IDE. O Selenium WebDriver ajuda a criar scripts de automação de testes mais complexos e avançados. Além disso, os testadores podem escrever em várias linguagens de programação como Java, Perl, JavaScript, PHP, Python, C#, Ruby e Groovy.

TestComplete: Plataforma comercial que automatiza e realiza testes em softwares para web, mobile e também desktop (Empresa: SmartBear Software). Permite também a utilização de diversas linguagens, como JavaScript, VBScript e Python, além também de ter as funções de teste orientado por dados (DDT), teste por palavras-chaves, teste de regressão e teste distribuído. Possibilita a criação de testes automatizados para aplicativos iOS, Web Microsoft Windows e Android. Comporta pelas seguintes funções:

- Testes de GUI;
- Reconhecimento de objetos;
- Atualização de interfaces do usuário automática;
- Suporte para diversas linguagens e tecnologias como: JavaScript, Python, VBScript, JScript, DelphiScript, C++Script e C#Script;
- Visualizador de testes;
- Testes com scripts;
- Gravação e reprodução de testes.

Telerik Test Studio: Uma ferramenta de automação abrangente de teste de software. Oferece vários tipos de testes, podendo ser usada manualmente. Oferece: teste funcional de UI, teste de performance, teste exploratório, teste mobile, teste de carregamento e teste no Visual Studio. Pode ser aplicada a diversas plataformas e a variadas linguagens (incluindo linguagens de script como VB.Net e C#). Compatível com aplicativos de automação como: Angular, ASP.NET, HTML5, JavaScript, AJAX, WPF, SilverLight, MVC, Ruby e iOS, PHP e Android. Funções principais:

- Gravar e reproduzir;
- Integração com Visual Basic Studio 2010 e 2012;
- Testes Cross-Browser;
- Integração com ferramentas de depuração;
- Testes manuais.

Obs.: *Sua licença é comercial e tem integração com vários sistemas e ferramentas para detectar erros.*

Robotium: Framework popular para a automação de testes focados em sistemas Android, de alta performance e é voltado para testes de interface gráfica. Benefícios:

- Código aberto;
- Compatível com aplicações nativas e híbridas;
- Facilita a escrita da automação de testes de caixa-preta;
- Boa compatibilidade com Gradle, Ant e Maven;
- API simples, todos os outros métodos estão disponíveis em solo class;
- Auxilia na escrita de automação de testes de Delays e Timings automáticos;
- Não é necessário escrever códigos ao navegar de uma atividade para outra;
- Os casos de teste são robustos, já que o tempo de processamento está vinculado aos componentes da UI.

WatiR: Aplicativo de testes de código aberto para Ruby. É open source muito leve, usada especialmente para testes de aplicações web. Gigantes da tecnologia como Oracle, SAP e Facebook usam. Funções:

- Testa qualquer navegador e aplicação web, independente da linguagem;
- Pode ser utilizada para automatizar testes em aplicações para todos os navegadores;
- Pode ser usada para testes em links, formulários, botões e tempo de respostas;
- Possibilidade de Testes Cross-Browser;
- Compatível com outras ferramentas como Cucumber, RSpec e Test/Unit;
- Testa os botões, links, forms e tempo de resposta de páginas web.

HPE Unified Functional Testing: Anteriormente se chamava QuickTest Professional (QTP da HP – Hewlett Packard), agora conhecido como Unified Functional Testing (UFT). Ferramentas de automação de testes Cross-Platform de ponta, que proporciona teste funcional automatizado e teste de regressão. Usa VBScript como linguagem de script, para especificar procedimentos de testes. Características exclusivas:

- Automatizar testes em Web, PowerBuilder, Desktop, ActiveX, SAP, Delphi, Net, Flex, Java, Oracle, Siebel, Mobile, PeopleSoft, Stingray, Visual Basic, etc;
- Oferece uso simplificado para GUI;
- Altamente integrado com a HP ALM (ferramenta de gerenciamento de testes) e HP LoadRunner (ferramenta de teste de performance);
- Integração com Mercury Business Process Testing e Mercury Quality Center;
- Exclusivo reconhecedor inteligente de objetos (Smart Object Recognition);
- Mecanismo de tratamento de erros;
- Criação de parâmetros para objetos, pontos de verificação e tabelas de dados;
- Documentação automatizada.

RanoRex: Utilizada para teste de GUI, bastante ajustável para executar e automatizar testes. Oferece diversas ferramentas de automatização de testes para aplicações web, desktop e mobile. Diferentemente do Selenium, o RanoRex é de fácil instalação e uso para não-programadores. Funções especiais:

- Reconhecedor de GUI;
- Código de testes reutilizáveis;
- Integração com diversas ferramentas;
- Permite gravar e reproduzir;
- Robusto identificador de objetos;
- Uso de expressões XPath;
- Edição XPath “Click & Go”;
- Editor de mapa de objetos de interface;
- Sincronização automática de objetos da UI.

Cucumber: Programado no Ruby, é uma ferramenta de testes open source desenvolvido com o conceito de desenvolvimento orientado por comportamento (BDD), conceito usado para escrever testes de aceitação para aplicações web. Anteriormente, era restrito apenas ao Ruby, mas atualmente suporta outras linguagens como Java, NET, Scala, Groovy, etc. Funcionalidades:

- Usado para performar testes automáticos de aceitação
- Fornece documentos únicos, que inclui especificações e testes
- Também suporta múltiplos sistemas operacionais.
- O código do Cucumber pode ser executado em diferentes frameworks como Selenium, Ruby, etc

Visual Studio Test Professional: Solução para testes totalmente instrumentada, configurável e intuitiva para todas as plataformas da Microsoft, mobile, tablets, desktops, servers e a nuvem. Benefícios:

- Com uma assinatura de MSDN você pode acessar outros produtos e serviços da Microsoft;
- Exploratory Browser Based Testing;
- É uma ferramenta licenciada útil para Streamlining Quality e Continuous Delivery;
- Também disponibiliza a opção free trial;
- Conduz, grava e repete testes manuais.

TestingWhiz: Ferramenta da empresa Cygnet Infotech para automação de testes, oferecendo soluções para testes em Website, teste de regressão, teste de aplicativos mobile, teste Cross-Browser, teste do banco de dados, testes de API’s. Funcionalidades importantes:

- Diferencial de Arquitetura sem código;
- Suporta bem integração contínua;

- Facilmente se integra a ferramentas de gerenciamento e de busca de erros;
- Testes guiados por palavra chave, dados ou distribuído;
- Grava e reproduz estruturas de testes automatizados;
- Object Eye Internal Recorder;
- Mais de 209 comandos de teste integrados agregados ao JavaScript;
- Integração com ferramentas de tracking de bugs como Jira, Mantis e Fogbugz;
- Integração com ferramentas de gerenciamento de testes como HP Quality Center;
- Testes baseados em risco;
- Integração e entrega contínua para ciclos ágeis.

Obs.: *Existem diversos pacotes que podem ser adquiridos conforme sua necessidade.*

Appium: Esta ferramenta de teste de software é open source e pode auxiliar em praticamente todas as linguagens de programação. Também pode operar testes em aplicações nativas, híbridas ou até mesmo multiplataforma. Outra vantagem dela é a automatização de testes simultâneos para os sistemas iOS e Android. Ainda, pode ser conectada à nuvem ou a um servidor local.

CEDUP Abílio Paulo – Criciúma – SC – *Tel.: (48)3438-0548/3403-1608/3403-1609*

Disciplina: **Teste de Software**

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Ferramentas de Teste de Velocidade de Site mais usadas

Fonte: As 20 Melhores Ferramentas de Teste de Velocidade de Site (hostinger.com.br – Acesso 03/11/2021).

Um site com desempenho ruim ou velocidade abaixo da esperada, é uma dor de cabeça para WebMaster. Isso, pode ser evitado, utilizando-se as métricas das ferramentas de teste de velocidade de site, que permitem monitorar a velocidade da sua página, como também, oferece dicas sobre melhorar os pontos fracos dela.

Por Que a Velocidade da Página Importa?

Você adora quando um site leva tempo demais para carregar?

"..., de acordo com o KissMetrics, 40% dos visitantes desistem de acessar um site se o processo de carregamento leva mais de três segundos."

O que afeta "a lentidão" no seu site:

1. ***Afeta a SERP*** — Um mecanismo de busca tem apenas um pequeno tempo para obter os resultados. Se a sua página não carrega dentro de uma curta janela de tempo, você terá um ranqueamento pior na lista da página de resultados do mecanismo de pesquisa (SERP).
2. ***Afeta o tráfego do site*** — Isso é o que nós queremos mostrar com aquele exemplo de antes. Um processo de carregamento mais lento significa um risco de 40% de perder visitantes em potencial (ou pior, clientes).
3. ***Afeta o SEO*** — Sites de busca têm a sua própria reputação para manter e a sua lenta velocidade de carregamento pode prejudicar a experiência de usuário deles.

Solução: Elevar a performance da sua página num bom nível para proporcionar a melhor experiência de usuário possível com um check-up periódico usando ferramentas de teste de velocidade. Abaixo, relacionado estão algumas ferramentas de Teste de Velocidade de Página, que servirão para diagnosticar e resolver os problemas de carregamento do seu site:

1. GTMetrix: Teste de velocidade de site GTMetrix – Na hora de lidar com a otimização de velocidade e de performance do seu site, os principais recursos do GTmetrix são adequados para quase todas as ocasiões. Essa ferramenta oferece um resumo dos principais indicadores de performance, do monitoramento de site e a habilidade de testar o seu site a partir de múltiplas regiões ao redor do mundo. Tudo de graça. Você também pode conduzir um teste de Throttling (redução de velocidade de conexão) para ver como o seu site está sendo carregado em diversas larguras de banda.

2. WebPageTest: Ferramenta de carregamento de Site WebPageTest – A ferramenta de teste de velocidade WebPageTest permite que você conduza um teste de velocidade a partir de múltiplos locais através do mundo, usando navegadores como Google Chrome e Internet Explorer de graça. Os seus recursos incluem testes de transação com múltiplos passos, captura de vídeo e bloqueio de conteúdo. No final do teste, você terá tabelas em cascata do carregamento de recursos, checagens de otimização de velocidade das páginas e sugestões para melhorias.

3. Google PageSpeed Insights: Teste de velocidade Google PageSpeed Insights – Como a Google é a criadora deste teste de velocidade, as métricas de experiência de usuário são baseadas na performance do site no Relatório do Chrome UX, tanto em dispositivos móveis quanto em desktop. O teste trará dados de laboratório e de campos. Isso inclui problemas de desempenho e dados de performance em tempo real sobre a experiência de uso dos seus usuários. Essa ferramenta de teste de velocidade é a escolhida por negócios de pequena e média escala – assim como donos de sites independentes que procuram por uma fácil, simples e direta de manter a performance do seu site.

4. Site Speed (Google Analytics): Recursos de análise site Speed do Google Analytics – Como parte do Google Analytics, o Site Speed avalia o desempenho do seu site com base em três aspectos: tempo de carregamento das páginas, velocidade de execução e duração da análise. O relatório de teste contém uma análise detalhada das páginas individuais e dos desempenhos dos recursos, assim como dicas de otimização customizadas.

5. Google Test My Site – Teste de velocidade Google Test My Site – Conforme SmartPhones vão se tornando mais comuns hoje em dia, os sites para dispositivos móveis precisam manter o desempenho em comparação com aqueles para desktop. O Test My Site mede o desempenho do site para mobile, promove BenchMarks contra seus competidores e oferece um relatório customizado junto de sugestões de como melhorar o desempenho das suas páginas.

6. Yslow – Recurso de análise de carregamento Yslow – Este projeto gratuito e de código aberto é uma ferramenta de teste de velocidade que analisa a performance do site com base em 23 das 34 regras do Yahoo! para sites de alta performance. Ele vem na forma de plugins para navegadores e scripts de linha de comando para servidores Node.js e PhantomJS. O Firefox é o navegador onde o Yslow foi implementado originalmente e isso significa que ele permite acesso total para os componentes de informação da página, através do Firebug Net Panel.

7. Pingdom – Teste de velocidade de Site Pingdom – Não apenas o Pingdom conduz um monitoramento minucioso do seu site, como ele também monitora as quedas. Ele usa mais de 70 locais de análise globais para testar as páginas da web. No final do teste, você receberá insights de performance e serão identificados os gargalos que atrapalham a velocidade de carregamento da sua página. Se você quiser um monitoramento ainda mais próximo, você pode comprar os planos pagos deles e ter outros serviços como monitoramento de uptime, verificação de velocidade da página, checagem de transações, insights sobre visitantes e acompanhamento de servidores. Seja qual for o problema, o recurso de alerta permitirá que você saiba imediatamente sobre ele.

8. KeyCDN Website Speed Test – Ferramenta de verificação KeyCDN – Com a opção de conduzir um teste de velocidade de 14 locais diferentes, os check-ups online do KeyCDN servem como uma ferramenta muito prática, que você pode executar até mesmo do seu próprio SmartPhone. Além de um teste de velocidade de página completo e de checagem de geolocalização, a ferramenta também pode executar um teste de ataque SSL FREAK para garantir a segurança do SSL/TLS do seu site.

9. DotCom-Monitor – teste de velocidade de site DotCom Tools – As ferramentas gratuitas do DotCom permitem que você conduza testes de tempo de carregamentos baseados em navegadores mobile e para desktop, através de 20 locais diferentes do planeta. O ponto positivo das ferramentas DotCom é o fato de que todos os testes geográficos podem ser realizados ao mesmo tempo. No final das contas, você receberá um relatório de desempenho individual e um relatório do tipo cascata de cada localização.

10. DareBoost – checagem de sites DareBoost – A ferramenta de teste de velocidade do DareBoost é capaz de conduzir monitoramento de performance a partir de 13 locais de testes e sete tipos de dispositivos diferentes – incluindo opções mobile. Os principais recursos dessa ferramenta incluem a habilidade de simular um teste de velocidade com e sem AdBlock e bloquear domínios específicos para descobrir o culpado do seu desempenho web abaixo do esperado. O teste produz um relatório detalhado junto de recomendações que são divididas entre diversas categorias. Esses agrupamentos fazem com que seja mais fácil priorizar as melhorias necessárias.

11. Geek Flare – Teste de velocidade de site com a ferramenta Geek Flare – O Geek Flare oferece monitoramento de desempenho padrão como qualquer outra ferramenta de teste de velocidade. Você pode testar a velocidade de carregamento do seu site em dispositivos desktop e mobile a partir de diversas localizações ao redor do mundo. As métricas de testes incluem tamanho da página, capturas de tela, contagens de solicitações por time e Tempo para o Primeiro Byte (TTFB).

12. New Relic – Teste de velocidade de site New Relic – Como uma empresa de software analítico, a New Relic conhece bem o setor de aplicativos de medição de performance. O seu teste sintético online permite que você verifique o seu desempenho na web a partir de nove regiões diferentes. Se você precisar de um teste mais amplo, eles têm planos pagos que são capazes de monitorar o seu sistema de maneira dinâmica, simulando comportamentos para isolar o problema, assim como analisar o impacto da sua performance no seu negócio.

13. LoadImpact – Teste de velocidade de site Load Impact – Essa ferramenta de testes baseada em nuvem se especializa em diagnosticar problemas de desempenho em sites na web, em App's e em API's. Usando o

recurso de testes k6 – que é de código aberto e baseado em linha de comando – os problemas podem ser detectados facilmente dentro do ciclo de desenvolvimento do software. Enquanto o teste de velocidade de página está disponível gratuitamente, você precisa comprar o plano deles para fazer uso dessas ferramentas adicionais.

14. Web Page Analyzer – Recurso de análise web site Optimization – Esta ferramenta gratuita traz um cálculo do tamanho da página, sua composição, o tempo de download e o tamanho de componentes individuais presentes nele. Então são feitas recomendações customizadas com base em dados como diretrizes gerais de tamanho de página assim como tendências e métodos de otimização de site.

15. Image Analysis Tool (Cloudinary) - Teste de velocidade de site Cloudinary – Se o seu site contém muitas imagens, a velocidade de carregamento pode acabar sendo prejudicada. Caso você se identifique com isso, recomendamos essa ferramenta do Cloudinary para avaliar problemas relacionados a imagens como tamanho, formato, qualidade e codificação.

16. Monitis – Verificação de tempo de carregamento Monitis – A ferramenta de testes gratuita do Monitis verifica o tempo de carregamento de todos os elementos do seu site. O teste é conduzido a partir dos EUA, da Europa e da Ásia de maneira simultânea. Com os seus planos pagos, você pode ter um escopo maior de monitoramento, que inclui o seu site completo, sua rede, seu servidor e seu aplicativo. Você também pode conduzir monitoramento customizado no seu sistema e verificar métricas de negócios usando API's.

17. Chrome DevTools – Teste de velocidade de site Chrome DevTools – Aqui está outra ferramenta da Google que pode ajudar a melhorar o tempo de carregamento do seu site. Com desenvolvedores como seus principais usuários, essas ferramentas facilitam o processo de edição em tempo real e o diagnóstico de problemas. A ferramenta é incorporada diretamente no navegador Chrome. Sua página inicial oficial oferece tutoriais para iniciantes que querem começar a mexer com programação simples e traz dicas para otimização de sites.

18. GiftOfSpeed – Recurso de análise de site GiftOfSpeed – Avalie o tempo de carregamento do seu site a partir de oito localidades diferentes usando o teste de velocidade do GiftOfSpeed. Você também pode melhorar o seu desempenho na web diretamente usando as outras ferramentas gratuitas como teste de otimização do CSS, teste de solicitações quebradas e compressor JavaScript.

19. UpTrends – Teste de velocidade com a ferramenta UpTrends – O UpTrends oferece uma ferramenta de teste de velocidade gratuita que avalia o tempo de carregamento do seu site no desktop ou em dispositivos móveis a partir de dez locais diferentes. Você também pode definir um Throttling de largura de banda e escolher no qual o teste será conduzido. A ferramenta de monitoramento de site também está disponível gratuitamente. Seus recursos incluem uma grande quantidade de locais de testes, alerta de e-mails e painéis de monitoramento.

20. BatchSpeed – Teste de velocidade BatchSpeed – Incorporando a API Google PageSpeed, a ferramenta BatchSpeed se especializa em realizar CrawLing de sites, de múltiplas URL's e de mapas de sites em XML antes de conduzir o teste de velocidade em si. O resultado pode ser organizado por velocidade, tamanho, recomendações ou níveis de prioridade.

CEDUP Abílio Paulo – Criciúma – SC – *Tel.: (48)3438-0548/3403-1608/3403-1609*

Disciplina: **Teste de Software**

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Sites Recomendados

Sites Brasileiros

- Softex
  http://www.softex.br/
- TestExpert
  http://www.testexpert.com.br/
- Associação Brasileira de Melhoria em Tecnologia da Informação
  http://www.abramti.org.br/
- Qualidade de Software
  http://qualidadesoftware.org.br/
- Qualidade de Software
  http://qualidade-de-software.blogspot.com/

Certificações

- Certificação Brasileira de Teste de Software (CBTS)
  http://www.alats.org.br/Default.aspx?tabid=198
- Brazilian Software Testing Qualification Board
  http://www.bstqb.org.br/
- Instituto Brasileiro de Qualidade em Testes de Software
  http://www.ibqts.com.br/
- Certified Software Quality Analyst (CSQA)
  http://www.softwarecertifications.org/
- Software Quality Engineer Certification CSQE
  http://www.asq.org/certification/software-quality-engineer/index.html
- ISEB Foundation Certificate in Software Testing
  http://www.bcs.org/server.php?show=nav.7179
- ISEB Practitioner Certificate in Software Testing
  http://www.bcs.org/server.php?show=nav.6956
- The Certified Software Tester (CSTE)
  http://www.softwarecertifications.org/
- Certified Test Manager (CTM)
  http://www.testinginstitute.com/ctm.php
- Certified Software Test Professional (CSTP)
  http://www.testinginstitute.com/cstp.php
- American Software Testing Qualifications Board
  http://www.astqb.org/

Revistas

- StickyMinds Software Quality & Test
  http://www.stickyminds.com/
- Software Test & Performance Magazine
  http://www.stpmag.com/
- SDTimes Magazine
  http://www.sdtimes.com/index.html
- The Rational Edge
  http://www-128.ibm.com/developerworks/rational/rationaledge/
- Professional Tester Magazine
  http://www.professionaltester.com/
- Free Software Magazine for the Free Software World
  http://www.freesoftwaremagazine.com/

CEDUP Abílio Paulo – Criciúma – SC – *Tel.: (48)3438-0548/3403-1608/3403-1609*

Disciplina: **Teste de Software**

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Fóruns de discussões

- VV-SW-Brasil - Validação e Verificação de Software
  http://br.groups.yahoo.com/group/VV-SW-Brasil/
- ALATS - Associação Latino Americana de Teste de Software
  http://br.groups.yahoo.com/group/alats-br/
- DFTestes: Grupo de amigos profissionais em Teste de Software do Distrito Federal
  http://br.groups.yahoo.com/group/DFTestes/
- QAI - Quality Assurance Institute Brasil
  http://br.groups.yahoo.com/group/qai-brasil/
- International Software Testing Qualifications Board
  http://groups.google.com.br/group/bstqb
- Grupo de estudo criado para quem deseja certificação CSTE da QAI
  http://br.groups.yahoo.com/group/cste-brasil/
- Grupo de discussão sobre Qualidade de Software
  http://br.groups.yahoo.com/group/qa_rs/
- QA Forums - The most popular Software Testing and QA discussions
  http://qaforums.com/
- Google Software Testing Group
  http://groups.google.com/group/SoftwareTesting
- Yahoo Software Testing and Quality Assurance Group
  http://groups.yahoo.com/group/Software_QA/
- Yahoo SQA Career Group
  http://groups.yahoo.com/group/SQACareer/
- Yahoo SQA Tester Group
  http://groups.yahoo.com/group/SQAtester/
- MSDN Software Testing Discussion
  http://forums.microsoft.com/msdn/showforum.aspx?forumid=1600&siteid=1

Ferramentas de Automação e Gestão de Testes

- Software QA Testing and Test Tool Resources
  http://www.aptest.com/resources.html
- Web Site Test Tools and Site Management Tools
  http://www.softwareqatest.com/qatweb1.html
- Open Source Software Testing Tools, News and Discussion
  http://opensourcetesting.org/
- Open Source Testing Tools in Java
  http://java-source.net/open-source/testing-tools
- Software QA and Testing Resource Center
  http://www.softwareqatest.com/qattls1.html

Outros

- Testing Spot
  http://testingspot.net/
- Search Software Quality
  http://searchsoftwarequality.techtarget.com/
- International Institute for Software Testing
  http://www.testinginstitute.com/
- Community portal for CSQA and CSTE professionals
  http://csqa.info/
- Food for Thought
  http://www.swqual.com/newsletter/Subscribe.htm

- Artima Weblogs
  http://www.artima.com/weblogs/index.jsp
- DevBistro Test Jobs
  http://www.devbistro.com/jobs/keywords/Test
- QA Jobs
  http://www.qajobs.net/
- Software Testing Hotlist
  http://www.io.com/~wazmo/qa/
- QAInsight
  http://qainsight.net/
- iSix Sigma Software/IT
  http://software.isixsigma.com/
- Quality Tree
  http://www.qualitytree.com/
- DevelopSense
  http://www.developsense.com/
- Satisfice
  http://www.satisfice.com/
- Developer Testing
  http://www.developertesting.com/
- Methods & Tools - Providing practical and free knowledge for the software developer, tester and project manager
  http://www.methodsandtools.com/
- Center for Software Testing Education
  http://www.testingeducation.org/BBST/
- TestFocus
  http://www.testfocus.co.za/
- Cem Kaner's website
  http://www.kaner.com/articles.html

Fonte: TestExpert

REFERENCIAS

https://www.monitoratec.com.br

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Anexos

I – GLOSSÁRIO – TESTE DE SOFTWARE

(Fonte: http://qualidade-de-software.blogspot.com/p/glossario-padrao-de-termos-utilizados.html).

GLOSSÁRIO PADRÃO DE TERMOS UTILIZADOS EM TESTE DE SOFTWARE

Versão 2.1br (Abril de 2010)
Produzido pelo "Glossary Working Party"
International Software Testing Qualification Board
Editor Erik van Veenendaal (Holanda)

A

Abordagem de teste *Test approach* Implementação da estratégia de teste para um projeto específico. Normalmente, inclui as decisões tomadas e baseadas no objetivo do projeto (teste) e na avaliação do risco feita, nos pontos de inicio relacionados ao processo de teste, nas técnicas de modelagem de teste a serem aplicadas, nos critérios de saída e nos tipos de testes a serem desempenhados.

Ação (IDEAL) *Acting (IDEAL)* A fase dentro do modelo IDEAL, onde as melhorias são desenvolvidas, postas em prática, e implementadas em toda a organização. A fase consiste nas atividades: criar solução, piloto/teste da solução, refinamento da solução e implementação da solução. *Ver também IDEAL.*

Aceite *Acceptance* Ver teste de aceite.

Acompanhamento *Walkthrough* Apresentação passo-a-passo feita pelo autor de um documento a fim de reunir informações e de estabelecer um entendimento comum sobre o seu conteúdo. [Freedman e Weinberg, IEEE 1028]. *Ver também revisão por pares.*

Acompanhamento estruturado *Structured walkthrough* Ver acompanhamento.

Acurácia *Accuracy* Capacidade do produto de software de prover, com o grau de precisão necessário, os resultados ou efeitos corretos ou acordados. [ISO-9126] *Ver também teste de funcionalidade.*

Adaptabilidade *Adaptability* Capacidade do produto de software de ser adaptado para diferentes ambientes, sem a necessidade de ações ou meios, além daqueles definidos pelo próprio software considerado. [ISO 9126] *Ver também portabilidade.*

Adequação *Suitability* Capacidade que um produto de software tem de fornecer um conjunto apropriado de funções para as tarefas especificadas e os objetivos do usuário. [ISO 9126] *Ver também funcionalidade.*

Alvo de teste *Test target* Conjunto de critérios de saída.

Ambiente de teste *Test environment* Ambiente que contém hardware, instrumentação, simuladores, ferramentas de software e outros elementos de suporte necessários à realização de um teste. [posterior a IEEE 610]

Ambiente operacional *Operational environment* Produtos de software ou hardware instalados nos locais de trabalho, residência dos usuários ou consumidores, onde o componente ou sistema sendo testado será utilizado. O software pode incluir sistemas operacionais, sistemas de gerenciamento de banco de dados e outros aplicativos.

Ambiente preparado para teste *Test harness* Ambiente de teste composto de simuladores e controladores necessários para a condução de um teste.

Analisabilidade *Analyzability* Capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas. [ISO 9126] *Ver também manutenibilidade.*

Analisador *Analyzer* Ver analisador estático.

Analisador de código estático *Static code analyzer* Ferramenta que realiza análise estática de código. A ferramenta checa: código-fonte, certas propriedades tais como a conformidade com os padrões de codificação, métricas de qualidade ou anomalias de fluxo de dados.

Analisador estático *Static analyzer* Ferramenta que realiza análise estática.

Análise causal *Causal analysis* Análise de defeitos para determinar a causa raiz. [CMMI]

Análise de causa-efeito *Cause-effect analysis* Ver gráfico de causa-efeito.

Análise de causa-raiz *Root cause analysis* Técnica de análise que visa identificar as causas dos defeitos. Ao orientar as medidas corretivas para as causas raiz, espera-se que a probabilidade de reincidência do defeito seja minimizada.

Análise de cobertura *Coverage analysis* Medição da cobertura alcançada por um item de cobertura específico durante a execução do teste com relação aos critérios pré-determinados, feita para determinar a necessidade de testes adicionais, e quais casos de teste seriam necessários.

Análise de código *Code analyzer* Ver análise de código estático.

Análise de código estático *Static code analysis* Análise do código-fonte realizada sem a execução desse software.

Análise de fluxo de dados *Data flow analysis* Forma de análise estática baseada na definição e uso de variáveis.

Análise de impacto *Impact analysis* Avaliação de mudança para as camadas de documentação de desenvolvimento, documentação, teste e componentes, a fim de implementar uma alteração dada aos requisitos especificados.

Análise de mutação *Mutation analysis* Método que determina a acuidade da suite de teste medindo a extensão até a qual uma suite de teste pode discernir entre o programa e suas pequenas variantes (mutantes).

Análise de Pareto *Pareto analysis* Técnica estatística para tomada de decisão que é usada na seleção de um número limitado de fatores que produzem efeito global significativo. Em termos de melhoria da qualidade, a grande maioria dos problemas (80%) são produzidos por algumas poucas causas essenciais (20%).

Análise de perigo *Hazard analysis* Uma técnica usada para caracterizar os elementos de risco. O resultado de uma análise de risco irá conduzir os métodos utilizados para o desenvolvimento e teste de um sistema. *Ver também análise de risco.*

Análise de Ponto de Função (FPA) *Function point analysis (FPA)* Método que visa medir o tamanho da funcionalidade de um sistema de informações. A medição independe da tecnologia. Ela pode ser utilizada como base para medição de produtividade, para a estimação dos recursos necessários e para controle de projeto.

Análise de Ponto de Teste (TPA) *Test point analysis (tpa)* Método de estimação de teste que usa fórmula baseada na Análise de Ponto de Função. [TMap]

Análise de risco *Risk analysis* Processo de avaliação dos riscos identificados para estimar o seu impacto e probabilidade de ocorrência (plausibilidade).

Análise de valor limite *Boundary value analysis* Técnica de projeto de teste caixa-preta onde os casos de teste são projetados com base nos valores da fronteira. *Ver também valor de fronteira.*

Análise dinâmica *Dynamic analysis* Processo de avaliação do comportamento. Por exemplo, o desempenho da memória ou o uso da CPU de um sistema ou componente durante sua execução. [posterior a IEEE 610]

Análise do fluxo de controle *Control flow analysis* Forma de análise estática com base em uma representação de caminhos únicos (sequência de eventos) na execução através de um componente ou sistema. A análise de fluxo de controle avalia a integridade das estruturas de controle de fluxo, procurando controlar possíveis anomalias, tais como loops ou etapas do processo logicamente inacessíveis.

Análise estática *Static analysis* Análise dos artefatos de software, por exemplo, requisitos ou código, realizado sem a execução desses artefatos de desenvolvimento de software. A análise estática é feita geralmente por meio de uma ferramenta de apoio.

Análise transacional *Transactional analysis* Análise das transações entre pessoas e mentes das pessoas, uma transação é definida como um estímulo acrescido de uma resposta. Transações ocorrem entre pessoas e entre os estados de ego (segmentos de personalidade) dentro da mente de uma pessoa.

Anomalia *Anomaly* Qualquer condição que se desvie da expectativa proposta pelas especificações dos requisitos para uma modelagem dos documentos, documentos de modelagem, padrões, etc. ou da percepção ou experiência de uma determinada pessoa. Dentre outras atividades, as anomalias podem ser encontradas durante as revisões, os testes, as análises, as compilações ou a aplicação do uso dos produtos de software ou da documentação. [IEEE 1044] *Ver também defeito, desvio, erro, dano, falha, incidente e problema.*

Aperfeiçoador do processo de teste *Test process improver* Pessoa que implementa melhorias no processo de teste baseado em um plano de melhoria de teste.

Apreensibilidade *Learnability* Capacidade que um produto de software tem de possibilitar ao usuário aprender suas aplicações. [ISO 9126] *Ver também usabilidade.*

Aprendizado (IDEAL) *Learning (IDEAL)* Fase dentro do modelo IDEAL onde se aprende com experiências e propõe melhorias e de adotar novos processos e tecnologias no futuro. A fase de aprendizagem consiste nas atividades: analisar, validar e propor ações futuras. *Ver também IDEAL.*

Aprovação de teste *Test pass* Ver aprovação.

Aprovação/reprovação de critérios *Pass/fail criteria* Regras de decisão usadas para determinar se um item de teste (função) ou recurso foi aprovado ou reprovado no teste. [IEEE 829]

Aprovação *Pass* Um teste é considerado aprovado se o seu resultado real coincide com o seu resultado esperado.

Armazenamento *Storage* Ver utilização de recurso.

Arranjo ortogonal *Orthogonal array* Matriz bidimensional construída com propriedades especiais de matemática, de tal forma que a escolha de qualquer par de colunas no arranjo fornece todos os pares de combinação de cada número do arranjo.

Árvore de classificação *Classification tree* Estrutura em árvore que mostra as partições de equivalência hirarquicamente ordenadas, usadas na modelagem de casos de teste e no método de classificação por árvore. *Ver também método de classificação por árvore.*

Ataque *Attack* Tentativa direcionada e focada de avaliar a qualidade, especialmente a confiabilidade, de um objeto de teste tentando forçar a ocorrência de falhas específicas. *Ver também testes negativos.*

Ataque ao software *Software attack* Ver ataque.

Ataque de falha *Fault attack* Ver ataque.

Atratividade *Attractiveness* Capacidade do produto de software de ser atraente ao usuário. [ISO 9126] *Ver também usabilidade.*

Atributo de qualidade *Quality attribute* Característica que afeta a qualidade de um item. [IEEE 610]

Auditor líder *Lead assessor* Pessoa que conduz uma avaliação. Em alguns casos, por exemplo, CMMi e TMMi quando são realizadas avaliações formais, o autitor líder deve ser credenciado e treinado formalmente.

Auditoria *Audit* Avaliação independente dos produtos de software ou processos a fim de verificar a conformidade com padrões, diretrizes, especificações e/ou procedimentos baseados em critérios objetivos incluindo documentos que especificam: (1) A forma ou o conteúdo dos produtos a serem produzidos. (2) O processo pelo qual os produtos deverão ser produzidos. (3) Como a conformidade aos padrões e diretrizes deverá ser medida [IEEE 1028].

Auditoria de configuração *Configuration auditing* Função que verifica o conteúdo das bibliotecas de itens de configuração, por exemplo, padrões de conformidade. [IEEE 610]

Automação da execução de teste *Test execution automation* Utilização de um software (por exemplo, ferramentas de captura/recuperação) para controlar a execução de testes, a comparação entre os resultados reais e os esperados, o estabelecimento de precondições e outras funções de controle de teste e de relato.

Automatização de teste *Test automation* Utilização de software para desempenhar ou dar suporte às atividades de teste, por exemplo, gerenciamento de teste, modelagem de teste, execução de teste e verificação de resultados.

Avaliação *Evaluation* Ver Teste.

Avaliação de processo *Process assessment* Avaliação disciplinada dos processos de software de uma organização contra um modelo de referência. [posterior a ISO 15504]

Avaliação heurística *Heuristic evaluation* Técnica estática de teste de usabilidade que determina o atendimento da interface de um usuário aos princípios de uso reconhecidos (os assim chamados princípios "heurísticos").

Avaliador *Assessor* Pessoa responsável por criar/acompanhar as avaliações/relatórios do projeto; qualquer membro da equipe de avaliação.

B

Balanced scorecard *Balanced scorecard* Ferramenta de gerenciamento estratégico de *performance* que possibilita medir o quanto as atividades operacionais de uma empresa estão alinhadas com seus objetivos em termos de visão de negócio e estratégia. *Ver também dashboard corporativo e scorecard.*

Base de teste *Test basis* Todos os documentos a partir dos quais os requisitos de um determinado componente ou sistema podem ser inferidos. Documentação na qual os casos de testes estão baseados. Se um documento pode ser alterado somente por meio de procedimento formal, então a base de teste passa a se chamar base de teste congelada. [posterior a TMap]

Base de teste congelada *Frozen test basis* Documento para base de teste que só pode ser alterado por um processo formal de controle de alteração. *Ver também linha de base.*

Baseline *Baseline* Especificação ou produto de software formalmente revisado ou acordado que servirá como base para futuros desenvolvimentos, pondendo ser alterado apenas por meio de um processo formal de controle de mudança. [posterior a IEEE 610].

Bebugging *Bebugging* [Abbot] Ver semeamento de falha.

Bloco básico *Basic block* Sequência de uma ou mais sentenças executáveis consecutivas que não contêm desvios. *Nota*: Um nó em um fluxograma representa um bloco básico.

Boas práticas *Best practice* Metodologia ou prática inovadora que contribui para o aumento do desempenho de uma organização em um determinado contexto, normalmente reconhecida como "melhores práticas" por organizações parceiras.

Buffer *Buffer* Dispositivo ou área de armazenamento usado para armazenar dados temporariamente em diferentes taxas de fluxo de dados, tempo, ocorrência de eventos, quantidades de dados que podem ser tratados pelo dispositivo ou processos envolvidos na transferência ou no uso dos dados. [IEEE 610]

Bug *Bug* Ver defeito.

C

Caminho *Path* Sequência de eventos (por exemplo, instruções executáveis) de um componente ou sistema a partir de um ponto de entrada para um ponto de saída.

Caminho de fluxo de controle *Control flow path* Ver caminho.

Caminho inviável *Infeasible path* Caminho que não pode ser exercido por nenhum conjunto de valores possíveis de entrada.

Caminho viável *Feasible path* Caminho para o qual um conjunto de valores de entrada e condições faz com que ele exista para ser executado.

Caminho-dd *Dd-path* Caminho de execução (geralmente através de um gráfico que representa um programa, como um fluxograma) que não inclui todos os nós condicionais, como o caminho de execução entre duas decisões.

Capability Maturity Model (CMM) *Capability maturity model (CMM)*
Modelo estruturado de cinco níveis que descreve os principais elementos de um processo de software eficaz. O Modelo de Maturidade de Capacidade cobre as melhores práticas para planejamento, engenharia e gerenciamento da manutenção e desenvolvimento de software.
Ver também, Capability Maturity Model Integration (CMMI).

Capability Maturity Model Integration (CMMI) *Capability maturity model integration (CMMI)*
Modelo estruturado que descreve os elementos-chave de um eficaz desenvolvimento de um produto e do seu processo de manutenção. O Capability Maturity Model Integration abrange as melhores práticas de planejamento, engenharia e desenvolvimento de produtos de gestão e manutenção. O CMMI é o sucessor do CMM. *Ver também Capability Maturity Model (CMM).*

Característica *Feature* Atributo de um componente ou sistema especificado ou implícito na documentação de requisitos (por exemplo, restrições de confiabilidade, de uso ou de modelagem). [Posterior a IEEE 610]

Característica de produto de software *Software product characteristic* Ver atributo de qualidade.

Característica de qualidade *Quality characteristic* Ver atributo de qualidade.

Característica de qualidade de software *Software quality characteristic* Ver atributo de qualidade.

Carta *Charter* Ver carta de teste.

Carta de teste *Test charter* Declaração dos objetivos do teste e de possíveis idéias sobre como realizar os testes. As cartas de teste são usadas em testes exploratórios. *Ver também testes exploratórios.*

CASE *CASE* Acrônimo para Computer Aided Software Engineering (Engenharia de Software Apoiada por Computador).

Caso de teste *Test case* Conjunto de valores de entrada, precondições de execução, resultados esperados e pós-condições de execução desenvolvidas para um determinado objetivo ou condição de teste, tais como para exercitar o caminho de um determinado programa ou verificar o atendimento a um requisito especifico. [posterior a IEEE 610]

Caso de teste abstrato *Abstract test case* Ver caso de teste de alto nível.

Caso de teste bloqueado *Blocked test case* Um caso de teste que não pode ser realizado porque as pré-condições para sua execução não estão atendidas.

Caso de teste concreto *Concrete test case* Ver caso de teste de baixo nível.

Caso de teste de alto nível *High level test case* Caso de teste sem valores concretos (nível de implementação) para os dados de entrada e para resultados esperados. Utilizam operadores e as instancias dos valores reais ainda não estão definidas e/ou disponíveis. *Ver também caso de teste de nível baixo.*

Caso de teste de baixo nível *Low level test case* Caso de teste com valores concretos (nível de implementação) para os dados de entrada e resultados esperados. Os operadores lógicos de casos de teste de alto nível são substituídos por valores reais que correspondem aos objetivos dos operadores lógicos. *Ver também caso de teste de alto nível.* Caso de teste lógico *Logical test case* Ver caso de teste de alto nível. Caso de uso *Use case* Sequência de transações em um diálogo entre um ator e um componente ou sistema, com um resultado tangível, onde um ator pode ser um usuário ou qualquer coisa que possa trocar informações com o sistema.

CAST *CAST* Acrônimo para Computer Aided Software Testing (Testes de Software Apoiados por Computador). *Ver também automatização de teste.*

Categoria de risco *Risk category* Ver tipo de risco.

Causa-raiz *Root cause* Origem de um defeito, que, se removida, fará com que a ocorrência do tipo de defeito seja diminuída ou removida.

Cenário de teste *Test scenario* Ver especificação de procedimento de teste.

Certificação *Certification* Processo de confirmar se um componente, sistema ou pessoa está em conformidade com requisitos pré-determinados. O processo pode ser executado, por exemplo, através da aplicação de uma prova.

Ciclo de Deming *Deming cycle* Processo de quatro etapas na resolução de problemas, (planejar, fazer, verificar, agir), normalmente usado em melhorias de processos. [posterior a Deming]

Ciclo de teste *Test cycle* Execução do processo de teste contra um único release identificável do objeto de teste.

Ciclo de vida do software *Software lifecycle* Período de tempo que começa quando um produto de software é concebido e termina quando o software não está mais disponível para uso. O ciclo de vida do software, normalmente inclui as fases de: conceito, requisitos, concepção, execução, teste, instalação e verificação, operação e manutenção e, às vezes, a fase de aposentadoria. Note que estas fases podem sobrepor-se ou serem realizadas de forma iterativa.

Classe de equivalência *Equivalence class* Ver participação de equivalência.

Cobertura *Coverage* Grau, expresso como uma porcentagem, que indica o quanto um item de cobertura foi exercitado por uma suite de testes.

Cobertura da condição de decisão *Decision condition coverage* Porcentagem de todos os resultados de condições e de decisões, que foram exercitados por uma suite de teste. 100% de cobertura de condição de decisão implicam em ter, ao mesmo tempo, 100% de cobertura de condição e 100% de cobertura de decisão.

Cobertura de caminho *Path coverage* Porcentagem de caminhos exercitada por uma suite de teste. 100% de cobertura de caminho implicam em 100% de cobertura LCSAJ.

Cobertura de código *Code coverage* Método de análise que determina quais partes do software foi, ou não, executada (ou coberta) pela suite de testes. Por exemplo, cobertura de sentença, cobertura de decisão e cobertura de condição.

Cobertura de combinação de condição *Condition combination coverage* Ver cobertura de condição múltipla.

Cobertura de condição *Condition coverage* Percentual de resultados desde que tenham sido executadas por um conjunto de testes. Cobertura de 100% condição exige que cada condição em cada instrução de decisão será testada como verdadeiro e falso.

Cobertura de condição múltipla *Multiple condition coverage* Percentual de combinação de todos os resultados de condição simples dentro de uma sentença que tem sido executada por um conjunto de testes. 100% de cobertura condição múltipla implica em uma cobertura de 100% determinação condição.

Cobertura de condição múltipla modificada *Modified multiple condition coverage* Ver cobertura de determinação de condição.

Cobertura de decisão *Decision coverage* Percentual de resultados de decisão que foram exercitados por uma suíte de teste. 100% de cobertura de decisão implicam em ter, ao mesmo tempo, 100% de cobertura de desvios e 100% de cobertura de sentenças.

Cobertura de decisão de condição modificada *Modified condition decision coverage* Ver cobertura de determinação de condição.

Cobertura de desvio *Branch coverage* Porcentagem de desvios no código exercitado por uma suíte de teste. Isso significa que 100% de cobertura de desvio implicam em 100% de cobertura de decisão e também em 100% de cobertura de sentença.

Cobertura de determinação de condição *Condition determination coverage* Porcentagem de todos os resultados de condições únicas que afeta de modo independente um resultado de decisão exercitado por uma suite de caso de teste. 100% de cobertura de determinação de condição implicam em 100% de cobertura de condição de decisão.

Cobertura de fluxo de dados *Data flow coverage* Porcentagem de pares de definição-uso exercitada por uma suite de teste.

Cobertura de partição de equivalência *Equivalence partition coverage* Percentual das partições de equivalência que foram exercitadas por uma suite de teste.

Cobertura de sentença *Statement coverage* Porcentagem de sentenças executáveis que tenham sido exercidas por um conjunto de testes.

Cobertura de teste *Test coverage* Ver cobertura.

Cobertura de valor limite *Boundary value coverage* Percentual de valores de fronteira que foram executados por uma suite de teste.

Cobertura estrutural *Structural coverage* Medidas de cobertura baseadas na estrutura interna de um componente ou sistema.

Cobertura LCSAJ *LCSAJ coverage* Porcentagem de LCSAJs de um componente exercitado por um suíte de teste. 100% de cobertura LCSAJ implicam em 100% de cobertura de decisão.

Cobertura N-switch *N-switch coverage* Porcentagem de sequências de transições N+1 exercitadas por uma suite de teste. [Chow]

Código *Code* Instruções de computador e definições de dados expressos em uma linguagem de programação ou em um formulário de saída por um montador, compilador ou outros tradutores. [IEEE 610]

Código inacessível *Unreachable code* Código que não pode ser alcançado e que, portanto, não pode ser executado.

Código morto *Dead code* Ver código inacessível.

Coexistência *Co-existence* Capacidade que um software tem de coexistir com outro software independente num ambiente comum compartilhando os mesmos recursos. [ISO 9126] *Ver também portabilidade.*

Combinação de cobertura de condição de desvio *Branch condition combination coverage* Ver cobertura de condições múltiplas.

Comitê de controle de configuração *Configuration control board (CCB)* Grupo de pessoas responsável pela avaliação e aprovação ou desaprovação das alterações propostas para os itens de configuração, e para assegurar a implementação das mudanças aprovadas. [IEEE 610]

Comitê de controle de mudança *Change control board* Ver comitê de controle de coniguração.

Comparação de teste *Test comparison* Processo que identifica diferenças entre os resultados reais produzidos pelo componente ou sistema sendo testado e os resultados esperados para o teste. A comparação de teste pode ser desempenhada durante a execução do teste (comparação dinâmica) ou após sua execução.

Comparação dinâmica *Dynamic comparison* Comparação entre os resultados reais e esperados realizada durante a execução do software, por exemplo, por uma ferramenta de execução de teste.

Comparação pós-execução *Post-execution comparison* Comparação entre os resultados reais e os esperados, desempenhada após o software ser executado.

Comparador *Comparator* Ver comparador de teste.

Comparador de teste *Test comparator* Ferramenta de teste que faz a comparação automatizada de testes.

Compilador *Compiler* Ferramenta de software que traduz programas expressos em uma linguagem de alto nível em sua linguagem de máquina equivalente. [IEEE 610]

Complexidade *Complexity* Grau de dificuldade de entendimento, manutenção e verificação que uma modelagem e/ou estrutura interna apresenta. *Ver também complexidade ciclomática.*

Complexidade ciclomática *Cyclomatic complexity* Número de caminhos independentes percorridos em um programa. A complexidade ciclomática é definida como: L - 2P + N, onde: » L = o número de arestas/links em um gráfico » N = número de nós em um gráfico » P = número de partes desconectadas do gráfico (por exemplo, um gráfico chamado ou sub-rotina)

[posterior a McCabe]

Componente *Component* Menor parte do sistema que pode ser testado isoladamente.

Comportamento *Behavior* A resposta de um componente ou sistema a um conjunto de pré-condições e valores de entrada.

Comportamento co-dependente *Codependent behavior* Dependência emocional ou psicológica excessiva em outra pessoa, especificamente em tentar mudar o comportamento dessa pessoa (indesejável), apoiando-os a continuar esse comportamento. Por exemplo, em testes de software, reclamando atraso na entrega de teste e ainda desfrutar do "heroísmo" necessário trabalhar horas adicionais para ganhar tempo quando a entrega atrasada, reforçando a intempestividade.

Comportamento relacionado a tempo *Time behavior* Ver desempenho.

Condição *Condition* Expressão lógica que pode ter como resposta "*verdadeiro* "ou "*falso*", como por exemplo, A > B. *Ver também condição de teste.*

Condição composta *Compound condition* Duas ou mais condições únicas, unidas por meio de um operador lógico. (AND, OR ou XOR), por exemplo, "A>B AND C>1000"

Condição de cobertura de desvio *Branch condition coverage* Ver cobertura de condição.

Condição de desvio *Branch condition* Ver condição.

Condição de saída *Condition outcome* Avaliação de uma condição em termos de verdadeiro ou falso.

Condição de teste *Test condition* Item ou evento de um componente ou sistema que pode ser verificado por um ou mais casos de teste, por exemplo, função, transação, característica, atributo de qualidade ou elemento estrutural.

Condição múltipla *Multiple condition* Ver condição composta.

Confiabilidade *Reliability* Capacidade do produto de software em executar suas funções exigidas sobcondições estabelecidas durante um determinado período de tempo, ou para um determinado número de operações. [ISO 9126]

Configuração *Configuration* Composição de um componente ou sistema definido pelo número, natureza e interconexões das partes que o constituem.

Conformidade *Compliance* Capacidade do produto de software de atender aos padrões, convenções ou regulamentações da lei e das prescrições similares [ISO 9126].

Conjunto de teste *Test set* Ver suite de teste.

Conjunto de testes base *Basis test set* Conjunto de casos de teste derivados da estrutura interna de um componente ou de uma especificação com o objetivo de assegurar que 100% de um determinado critério de cobertura seja alcançado.

Consequência do teste *Test outcome* Ver resultado.

Consequência prevista *Predicted outcome* Ver resultado esperado.

Consequência real *Actual outcome* Ver resultado real.

Consistência *Consistency* Grau de uniformidade, padronização e livre de contradição entre os documentos ou partes de um componente ou sistema. [IEEE 610]

Controlador *Driver* Um componente de software ou ferramenta de testes que substitui um componente que é responsável pelo controle e/ou chamada de um componente ou sistema. [posterior a Tmap]

Controlador de teste *Test driver* Ver controlador.

Controle de configuração *Configuration control* Elemento de gerenciamento de configuração que consiste na avaliação, coordenação, aprovação ou desaprovação e execução de alterações nos itens de configuração após a criação formal da sua identificação de configuração. [IEEE 610]

Controle de mudança *Change control* Ver controle de configuração.

Controle de risco *Risk control* Processo que toma decisões e implementa medidas de proteção para reduzir riscos ou para mantê-los em níveis específicos.

Controle de teste *Test control* Tarefa do gerenciamento de teste que lida como desenvolvimento e aplicação de um conjunto de ações corretivas quando o monitoramento mostra qualquer desvio do originalmente planejado. *Ver também gerenciamento de teste.*

Controle de versão *Version control* Ver controle de configuração.

COTS *COTS* Acrônimo para Commercial Off-The-Shelf Software (Software Comercial de Prateleira). Ver software de prateleira.

Critério de aceite *Acceptance criteria* Critérios de saída que um componente ou sistema deve satisfazer para ser aceito por um usuário, cliente ou outra entidade autorizada. [IEEE 610].

Critério de conclusão do teste *Test completion criteria* Ver critério de saída.

Critério de reinício *Resumption criteria* Atividades que devem ser repetidas quando um teste for reiniciado após sua suspensão. [posterior a IEEE 829]

Critério de suspensão *Suspension criteria* São os critérios utilizados para interromper (temporariamente) todas ou parte das atividades de teste nos itens de teste. [posterior a IEEE 829]

Critérios de conclusão *Completion criteria* Ver critérios de saída.

Critérios de entrada *Entry criteria* Conjunto de condições genéricas e específicas que permitem que um processo avance com uma determinada tarefa (por exemplo, fase de teste). A finalidade dos critérios de entrada é evitar que uma tarefa implique em mais esforços (desperdício) em comparação com o esforço necessário.

Critérios de saída *Exit criteria* Conjunto de condições genéricas e específicas, acordadas pelos stakeholders, que permite que um processo seja oficialmente considerado completado. A finalidade dos critérios de saída é evitar que uma tarefa seja considerada completa quando ainda existirem partes importantes dela que ainda não tenham sido terminadas. Os critérios de saída são utilizados para relatar e para planejar o momento de interromper os testes. [posterior a Gilb e Graham]

Cronograma de execução de teste *Test execution schedule* Esquema para a execução dos procedimentos de teste. Os procedimentos de teste são incluídos no cronograma do contexto de execução do teste na ordem em que deverão ser executados.

Cronograma de teste *Test schedule* Lista de atividades, tarefas e eventos do processo de teste, identificando o seu início previsto e datas de término e/ou tempos, e interdependências entre tarefas.

CTP *CTP* Ver Processos Críticos de Teste.

Custo da qualidade *Cost of quality* Custos totais incorridos em atividades de qualidade e em questões frequentemente divididas em custos de prevenção, custos de avaliação, os custos de falhas internas e custos de falhas externas.

D

Dados de teste *Test data* Dados existentes (por exemplo, em um banco de dados) antes do início da execução de um teste e que afetam ou são afetados pelo componente ou sistema sendo testado.

Dashboard corporativo *Corporate dashboard* Representação no estilo de painel de controle do status dos dados de *performance* da corporação. *Ver também Balanced Scorecard e dashboard.*

Decisão *Decision* Nome dado ao ponto de um programa no qual o fluxo de controle tem duas ou mais rotas alternativas. Um nó com dois ou mais links para separar os desvios.

Defeito *Defect* Falha em um componente ou sistema que pode fazer com que o componente ou sistema falhe ao desempenhar sua função (por exemplo, uma sentença incorreta ou uma definição de dados incorreta). Um defeito, se descoberto durante a execução, pode levar a falha do componente ou do sistema.

Definição de dados *Data definition* Sentença executável na qual é atribuído um valor a uma determinada variável.

Densidade de falha *Fault density* Ver densidade de defeito.

Densidade do defeito *Defect density* Número de defeitos identificados em um componente ou sistema dividido pelo tamanho do componente ou sistema, expresso em termos de medida padrão (por exemplo, linhas de código, número de classes ou pontos de função).

Depuração de código *Debugging* Processo de procurar, analisar e remover as causas de falhas no software.

Depurador *Debugger* Ver ferramenta de depuração de código.

Desempenho *Performance* Grau em que um sistema ou componente realiza suas funções designadas dentro das restrições dadas, quanto tempo de processamento e taxa de transmissão. [posterior a IEEE 610] *Ver também eficiência.*

Desenvolvimento ágil de software *Agile software development* Um grupo de metodologias de desenvolvimento de sistemas baseadas em interações incrementais onde os requisitos e soluções evoluem através da colaboração entre equipes de especialistas independentes e de diferentes áreas.

Desenvolvimento orientado ao teste *Test driven development* Modo de desenvolvimento de software no qual os casos de teste são desenvolvidos, e frequentemente automatizados, antes que o software seja desenvolvido para rodar esses casos de teste.

Desvio *Branch* Bloco básico que pode ser selecionado para execução baseado na construção de um programa no qual um ou mais caminhos alternativos estejam disponíveis (por exemplo, *case*, *jump*, *go to*, *if-then-else*).

Diagnóstico (IDEAL) *Diagnosing (IDEAL)* Fase dentro do modelo IDEAL quando se determina onde se está em relação a onde se quer estar. A fase de diagnóstico consiste das atividades: caracterizar o estado atual desejado e desenvolver recomendações. *Ver também IDEAL.*

Diagrama causa-efeito *Cause-effect diagram* Representação gráfica usada para organizar e visualizar as relações entre várias causas possíveis de um problema. As possíveis causas são organizadas em categorias e subcategorias na forma de uma estrutura em árvore horizontal, com o defeito (potencial) ou o falha como o nó raiz. [posterior a Juran]

Diagrama de estado *State diagram* Diagrama que descreve os estados que um componente ou sistema pode assumir. Mostra também os eventos e circunstâncias que causam e/ou resultam da alteração de um estado para outro. [IEEE 610]

Diagrama espinha de peixe *Fishbone diagram* Ver diagrama causa-efeito.

Diagrama Ishikawa *Ishikawa diagram* Ver diagrama causa-efeito.

Disponibilidade *Availability* Medida pela qual um componente ou sistema está operacional e acessível quando requisitado. Frequentemente é expresso em porcentagem. [IEEE 610]

Domínio *Domain* Conjunto a partir do qual valores válidos de entrada e/ou saída podem ser selecionados.

Domínio de entrada *Input domain* Conjunto a partir do qual os valores de entrada válidos podem ser selecionados. *Ver também domínio.*

Domínio de saída *Output domain* Conjunto a partir do qual valores de saída válidos podem ser selecionados. *Ver também domínio.*

E

Efeito de monitoração *Probe effect* Efeito causado no componente ou sistema pelo instrumento de medição quando o componente ou sistema está sendo medido, exemplo, por uma ferramenta de teste de desempenho ou por um monitor. Por exemplo, o desempenho poderá ser um pouco pior quando as ferramentas de teste de desempenho forem utilizadas.

Eficiência *Efficiency* Capacidade do sistema de fornecer um desempenho apropriado em relação à quantidade de recursos utilizados sob determinadas condições [ISO 9126]

Emulador *Emulator* Dispositivo, programa de computador ou sistema que aceita as mesmas entradas e produz as mesmas saídas de um dado sistema. [IEEE 610] *Ver também simulador.*

Engano *Mistake* Ver erro.

Entendibilidade *Understandability* Capacidade que um produto de software tem de possibilitar ao usuário entender se o software é adequado para uso, e como ele pode ser utilizado em determinadas tarefas e condições de uso. [ISO 9126]. *Ver também usabilidade.*

Entrada *Input* Variável (seja armazenado dentro ou fora de um componente) que é lida por um componente.

Entrada de teste *Test input* Dados recebidos pelo objeto do teste de uma fonte externa durante a execução do teste. A fonte externa pode ser um hardware, um software ou uma pessoa.

Entrada especificada *Specified input* Entrada para o qual a especificação prevê um resultado.

Entregáveis de teste *Test deliverable* Qualquer produto do teste que deve ser entregue a alguém que não seja o autor produto de teste. *Ver também entrega.*

Entregável *Deliverable* Qualquer produto que deva ser entregue a alguém que não ao proprio autor.

Equipamento de teste *Test rig* Ver ambiente de teste.

Erro *Error* Ação humana que produz um resultado incorreto. [posterior a IEEE 610]

Escala de medição *Measurement scale* Escala que restringe o tipo de análise de dados que pode ser desempenhada nela. [ISO 14598]

Escalabilidade *Scalability* Capacidade que um produto de software tem para sofrer um upgrade ou para acomodar aumento de cargas. [posterior a Gerrard]

Especificação *Specification* Documento que especifica, de preferência de forma completa, precisa e verificável, requisitos, projetos, comportamento ou outras características de um componente ou sistema, e, muitas vezes, os procedimentos para determinar se essas disposições foram satisfeitas. [posterior a IEEE 610]

Especificação de caso de teste *Test case specification* Documento que especifica um conjunto de casos de teste (objetivos, entradas, ações do teste, resultados esperados e precondições para execução) para um item de teste. [posterior a IEEE 829]

Especificação de componente *Component specification* Descrição da função de um componente em termos de seus valores de saída para valores de entrada específicos, sob situações especificas e descrição do comportamento não funcional requerido (por exemplo, utilização de recursos).

Especificação de modelagem de teste *Test design specification* Documento que especifica as condições de teste (cobertura de itens) para um item de teste. Detalha a abordagem de teste e identifica os casos de teste de alto nível associados. [posterior a IEEE 829]

Especificação de procedimento de teste *Test procedure specification* Documento que especifica uma sequência de ações para a execução de um teste. Também conhecido como script de teste ou script de teste manual. [posterior a IEEE 829]

Especificação de teste *Test specification* Documento que consiste em uma especificação de projeto de teste, do caso de teste e/ou do procedimento de teste.

Estabelecimento (IDEAL) *Establishing (IDEAL)* Fase dentro do modelo IDEAL aonde os detalhes de como uma organização chegará a sua meta são planejados. A fase de estabelecimento consiste nas atividades de definir prioridades, desenvolver a abordagem e planejar ações. *Ver também IDEAL.*

Estabilidade *Stability* Capacidade que um produto de software tem para evitar efeitos inesperados resultantes de modificações feitas em um software. [ISO 9126]. *Ver também mantenabilidade.*

Estágio de teste *Test stage* Ver nível de teste.

Estimativa de teste *Test estimation* Aproximação calculada de um resultado relacionado com vários aspectos do teste (por exemplo, esforço despendido, data de conclusão, custos envolvidos, número de casos de teste, etc), que é utilizável mesmo se os dados de entrada sejam incompletos, incertos ou incompreensíveis.

Estouro de buffer *Buffer overflow* Falha no acesso de memória devido ao processo de armazenamento de dados ultrapassar o limite do tamanho da área de armazenamento temporário, resultando em estouro das áreas de memória adjacente ou levantamento de excessão. *Ver também buffer.*

Estratégia de teste *Test strategy* Descrição de alto nível dos níveis de teste a serem realizados e do teste dentro desses níveis feitos para uma organização ou programa (um ou mais projetos).

Execução de teste *Test execution* Processo de executar um teste em um componente ou sistema sendo testado e que produz resultado(s) real(ais).

Exercitado *Exercised* Um elemento de programa é considerado exercitado por um caso de teste quando o valor de entrada causa a execução deste elemento (por exemplo, uma sentença, uma decisão ou outro elemento estrutural).

Extreme Programming *Extreme Programming* Metodologia de engenharia de software utilizada no desenvolvimento ágil de software em que as práticas fundamentais são a programação por pares, fazendo ampla revisão de código, testes de unidade de todo o código, e a simplicidade e clareza no código. *Ver também desenvolvimento ágil de software.* 2

F

Failure Mode and Effect Analysis (FMEA) *Failure Mode and Effect Analysis (FMEA)* Abordagem sistemática da identificação de risco e da análise da identificação de possíveis modos de falha e das tentativas de prevenção de sua ocorrência.

Failure Mode, Effects, and Criticality Analysis (FMECA) *Failure Mode, Effects, and Criticality Analysis (FMECA)* Extensão do FMEA, em adição à base do FMEA, que inclui uma análise de criticidade, que é usada para traçar a probabilidade de modos de falha em relação à gravidade das suas consequências. O resultado destaca modos de falha com probabilidade relativamente elevada e gravidade das consequências, permitindo dirigir esforços dirigidos de reparação, onde será produzido maior valor.
Ver também Failure Mode and Effect Analysis (FMEA).

Falha *Failure* Desvio do componente ou sistema da entrega, resultado ou serviço esperado. [posterior a Fenton]

Falhar *Fail* Um teste é considerado falho se o seu resultado real não corresponde ao resultado esperado.

Falso resultado aprovado *False-pass result* Resultado de teste que não consegue identificar a presença de um defeito que está presente no objeto de teste.

Falso resultado falho *False-fail result* Resultado de testes onde um defeito é aberto embora nenhum defeito exista no objeto do teste.

Falso resultado negativo *False-negative result* Ver falso resultado falho.

Falso resultado positivo *False-positive result* Ver falso resultado aprovado.

Fase de execução e teste *Test execution phase* Período de tempo do ciclo de vida de desenvolvimento de um software durante o qual os componentes de um produto de software são executados, e o produto de software é avaliado para determinar se os requisitos foram ou não satisfeitos. [IEEE 610]

Fase de requisitos *Requirements phase* Período de tempo no ciclo de vida do software durante o qual os requisitos para um produto de software são definidos e documentados. IEEE [610]

Fase de teste *Test phase* Conjunto distinto de atividades de teste coletadas em uma fase gerenciável do projeto, exemplo, durante a execução das atividades de um nível de teste. [posterior a Gerrard]

Fator crítico de sucesso *Critical success factor* Elemento necessário para que uma organização ou projeto consiga realizar sua missão. São fatores críticos ou atividades necessárias para assegurar o sucesso. *Ver também modelo com base em conteúdo.*

Fault Tree Analysis (FTA) *Fault tree analysis (FTA)*
Técnica utilizada para analisar as causas das falhas (defeitos). Modelo de técnica visual que apresenta as relações lógicas entre as falhas, erros humanos, e os eventos externos que podem se combinar para causar falhas específicas de divulgação.

Fechamento de teste *Test closure* Durante a fase de fechamento de um processo de teste, coletam-se dados das atividades já completadas a fim de consolidar a experiência, o testware, os fatos e os números. A fase de fechamento consiste em finalizar e arquivar o testware e em avaliar o processo de teste, inclusive com a preparação de um relatório de avaliação de teste. *Ver também processo de teste.*

Ferramenta de análise dinâmica *Dynamic analysis tool* Ferramenta que fornece informações em tempo de execução sobre o estado do código do software. Estas ferramentas são mais comumente usadas para identificar indicadores atribuídos, verificar funções aritméticas e monitorar a memória quanto à alocação, utilização, desalocação e vazamentos.

Ferramenta de análise estática *Static analysis tool* Ver analisador estático.

Ferramenta de bug tracking *Bug tracking tool* Ver ferramenta de gerenciamento de defeito.

Ferramenta de captura e execução *Capture/playback tool* Tipo de ferramenta de execução de teste onde os valores de entrada são gravados durante o teste manual a fim de gerar scripts de testes automatizados que possam ser executados mais tarde, ou seja, reproduzidos. Essas ferramentas são frequentemente utilizadas para apoiar testes de regressão automatizada.

Ferramenta de captura e reprodução *Capture/replay tool* Ver ferramenta de captura e execução.

Ferramenta de cobertura *Coverage tool* Ferramenta que fornece medidas objetivas de quais elementos estruturais (por exemplo, sentenças ou desvios) foram exercitados por uma suíte de teste.

Ferramenta de depuração de código *Debugging tool* Ferramenta utilizada por programadores para reproduzir falhas, investigar o estado dos programas e procurar defeitos. A ferramenta de depuração permite aos programadores executar programas passo a passo, interromper um programa em qualquer sentença, assim como definir e examinar variáveis de programação.

Ferramenta de execução de teste *Test execution tool* Tipo de ferramenta de teste que pode executar outro software utilizando um roteiro de teste automatizado, ex. captura/recuperação. [Fewster e Graham]

Ferramenta de gerenciamento de defeito *Defect management tool* Ferramenta que facilita a gravação, monitoramento e alterações de defeitos. Possuem frequentemente recursos orientados para o fluxo de trabalho a fim de rastrear e controlar a alocação, a correção e a nova realização de testes de defeitos, além de fornecerem

---

recursos para a elaboração de relatórios. *Ver também incident management tool (ferramenta de gerenciamento de incidentes).*

Ferramenta de gerenciamento de incidente *Incident management tool* Ferramenta que facilita o registro e o rastreamento de condição de incidentes. Frequentemente possuí recursos orientados para o fluxo de trabalho para rastrear e controlar a alocação, correção e nova realização de testes de incidentes, além de fornecer recursos para relatório. *Ver também ferramenta de gerenciamento de defeito.*

Ferramenta de gerenciamento de requisito *Requirements management tool* Ferramenta que suporta a gravação de requisitos, atributos de requisitos (por exemplo, prioridade, o responsável pelo conhecimento) e anotações, facilitando a rastreabilidade através de camadas de requisitos e gerenciamento das mudanças de requisitos. Algumas ferramentas de gerenciamento de requisitos também proporcionam meios de análise estática, como a verificação de consistência e violações de regras pré-definidas.

Ferramenta de gerenciamento de teste *Test management tool* Ferramenta que dá suporte ao gerenciamento de teste e que controla parte deste processo. Frequentemente possui várias capacidades, tais como, gerenciamento de testware, estabelecimento de um cronograma de testes, registro dos resultados, rastreamento do progresso, gerenciamento de incidentes e relato de teste.

Ferramenta de gestão de configuração *Configuration management tool* Ferramenta que dá suporte para identificação e controle dos itens de configuração, o estado durante as mudanças e versões e a liberação das linhas de base que fazem parte dos itens de configuração.

Ferramenta de gravação/recuperação *Record/playback tool* Ver ferramenta de captura e recuperação.

Ferramenta de medição de cobertura *Coverage measurement tool* Ver ferramenta de cobertura.

Ferramenta de modelagem *Modeling tool* Ferramenta que suporta a criação, alteração e verificação dos modelos de software ou sistema [Graham].

Ferramenta de modelagem de teste *Test design tool* Ferramenta que dá suporte à atividade de modelagem de teste por meio da geração de entradas de teste a partir de uma especificação que pode estar armazenada em um repositório de ferramenta CASE (por exemplo, ferramenta de gerenciamento de requisitos a partir de condições de teste especificadas armazenadas na ferramenta em si ou em um código).

Ferramenta de monitoramento *Monitoring tool* Ver monitor.

Ferramenta de preparação de dados de teste *Test data preparation tool* Tipo de ferramenta de teste que possibilita que os dados sejam selecionados dos bancos de dados existentes ou que sejam criados, gerados, manipulados e editados para uso no teste.

Ferramenta de rastreamento de defeito *Defect tracking tool* Ver ferramenta de gerenciamento de defeitos.

Ferramenta de revisão *Review tool* Ferramenta que dá suporte ao processo de revisão. Suas características normalmente incluem o planejamento da revisão e o suporte ao rastreamento, assim como suporte às comunicações, revisões colaborativas e um repositório para coletar e relatar as métricas.

Ferramenta de segurança *Security tool* Ferramenta que oferece suporte à segurança operacional.

Ferramenta de semeamento de falha *Fault seeding tool* Ferramenta para inserir intencionalmente (semear) falhas em um componente ou sistema.

Ferramenta de teste *Test tool* Produto de software que dá suporte a uma ou mais atividades de um teste, entre elas, planejamento e controle, especificação, construção de arquivos iniciais e dados, execução e análise de testes. [TMap]. *Ver também CAST.*

Ferramenta de teste de carga *Load testing tool* Ver ferramenta de teste de *performance*.

Ferramenta de teste de desempenho *Performance testing tool* Ferramenta que dá suporte ao teste de desempenho e que, normalmente, tem dois recursos principais de medição de geração de carga e de transação de teste. A geração de carga pode simular tanto os usuários múltiplos como os altos volumes de dados de entrada. Durante a execução, as medições dos tempos de resposta são feitas a partir de transações selecionadas e depois registradas. Normalmente, as ferramentas de teste de desempenho fornecem relatórios baseados nos registros e grafos de testes da carga em relação aos tempos de resposta.

Ferramenta de teste de estresse *Stress testing tool* Ferramenta que suporta teste de estresse.

Ferramenta de teste de hyperlink *Hyperlink test tool* Ferramenta utilizada para verificar se existem hyperlinks desfeitos (quebrados) presentes em uma página da web.

Ferramenta de teste de segurança *Security testing tool* Ferramenta que dá suporte para o teste das características de vulnerabilidades de segurança.

Ferramenta para semeamento de erro *Error seeding tool* Ver ferramenta para semeamento de falha.

Fluxo de controle *Control flow* Sequência de eventos (caminhos) na execução através de um componente ou sistema.

Fluxo de dados *Data flow* Uma representação abstrata da sequência e eventuais alterações do estado de objetos de dados, onde o estado de um objeto é qualquer um: criação, uso ou destruição. [Beizer]

Framework de teste de unidade *Unit test framework* Ferramenta que proporciona um ambiente de teste de unidade ou de componentes em que um componente pode ser testado de forma isolada ou com stubs e drivers adequados. Ele também fornece suporte para o desenvolvedor de outras, tais como capacidades de depuração. [Graham]

Funcionalidade *Functionality* Capacidade do produto de software de oferecer funções que atendam às necessidades declaradas ou implícitas quando utilizado sob condições específicas. [ISO 9126]

Funcionalidade de software *Software feature* Ver funcionalidade.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

G

Garantia de qualidade *Quality assurance* Parte do gerenciamento de qualidade que garante que os requisitos de qualidade sejam tendidos. [ISO 9000]

Gerador de teste *Test generator* Ver ferramenta de preparação de dados de teste.

Gerenciamento de defeito *Defect management* Processo de reconhecimento, investigação, tomada de medidas e eliminação de defeitos. Trata-se de gravação de defeitos, classificação e identificação do impacto. [posterior a IEEE 1044]

Gerenciamento de incidente *Incident management* O processo de reconhecimento, investigação, tomada de medidas e eliminação de incidentes. Trata-se de registrar os incidentes, classificando-os e identificando o impacto. [posterior a IEEE 1044]

Gerenciamento de mudança *Change management* (1) abordagem estruturada de transição de indivíduos, equipes e organizações a partir de um estado atual para um estado futuro desejado. (2) forma controlada para efetuar uma mudança, ou uma proposta de mudança, para um produto ou serviço. *Ver também gerenciamento de configuração.*

Gerenciamento de problema *Problem management* Ver gerenciamento de defeito.

Gerenciamento de qualidade *Quality management* Atividades coordenadas para dirigir e controlar a qualidade em uma organização. Direção e controle de qualidade geralmente incluem o estabelecimento da política e dos objetivos de qualidade, assim como o planejamento, o controle, a garantia e a melhoria da qualidade. [ISO 9000]

Gerenciamento de risco *Risk management* Aplicação sistemática de procedimentos e práticas de tarefas para identificar, analisar, priorizar e controlar o risco.

Gerenciamento de teste *Test management* Planejamento, estimação, monitoramento e controle das atividades de teste, normalmente conduzidos pelo gerente de teste. Gerenciamento de teste baseado em sessão *Session-based test management* Método para medir e gerenciar testes baseados em sessões, por exemplo, testes exploratórios.

Gerente de teste *Test manager* Pessoa responsável pelo gerenciamento do projeto, pelas atividades e recursos de teste e por avaliar o objeto de teste. É o indivíduo que dirige, controla, administra, planeja e regula a avaliação de um objeto de teste.

Gestão de configuração *Configuration management* Disciplina que aplica o monitoramento e as direções técnicas e administrativas para identificar e documentar as características funcionais e físicas de um item de configuração, controlar as mudanças destas características, armazenar e informar os processos de mudança.

Goal Question Metric *Goal Question Metric*
Abordagem para a medição de software através de um modelo em três níveis: nível conceitual (objetivo), nível operacional (causa) e nível quantitativo (métricas).

GQM *GQM* Ver Goal Question Metric.

Gráfico de causa-efeito *Cause-effect graph* Representação gráfica de entradas e/ou estímulos (causas) com suas saídas associadas (efeitos), os quais podem ser usados para projetar casos de testes.

Gráfico de chamada *Call graph* Representação abstrata dos relacionamentos de chamadas entre as subrotinas de um programa.

Grafico do fluxo de controle *Control flow graph* Representação abstrata de todas as possíveis sequências de eventos (caminhos) na execução de um componente ou sistema.

Gravação de teste *Test recording* Ver registrar teste.

Grupo de Processo de Teste *Test process group* Grupo de especialistas em teste que auxiliam na definição, manutenção e melhoria dos processos de teste utilizados pela organização. [posterior a CMMI]

Guia de instalação *Installation guide* Instruções fornecidas por qualquer mídia adequada que guiam uma pessoa durante o processo de instalação. Pode ser um guia manual, um procedimento passo a passo, um assistente de instalação ou qualquer outro processo similar.

H

Hyperlink *Hyperlink* Ponteiro dentro de uma página web que leva a outras páginas da web.

I

IDEAL *IDEAL* Modelo de melhoria organizacional que serve como um roteiro para a iniciação, planejamento e implementação de ações de melhoria. O modelo IDEAL é baseado em cinco fases: inicialização, diagnóstico, estabelecimento, ação e aprendizado.

Identificação da configuração *Configuration identification* Elemento de gerenciamento de configuração, que consiste em selecionar os itens de configuração de um sistema e gravar suas características funcionais e físicas em uma documentação técnica. [IEEE 610]

Identificação de risco *Risk identification* Processo que identifica os riscos por meio de técnicas como brainstorming, listas de verificação ou histórico de falhas.

Implementação de teste *Test implementation* Processo de desenvolvimento e priorização dos procedimentos de teste, criação de dados e, opcionalmente, preparando os equipamentos de teste e criando scripts de testes automatizados.

Incidente *Incident* Qualquer ocorrência de evento que requer uma investigação. [posterior a IEEE 1008]

Incidente de teste *Test incident* Ver incidente.

Incidente de teste de software *Software test incident* Ver incidente.

Independência do teste *Independence of testing* Separação das responsabilidades, o que incentiva a realização de testes objetivos. [posterior a 178B]

Indicador *Indicator* Medida que pode ser usada para estimar ou prever outra medida. [ISO 14598]

Indicador de desempenho *Performance indicator* Métrica de nível alto de eficácia e/ou eficiência utilizada para guiar e controlar o desenvolvimento progressivo, por exemplo, deslizes no acompanhamento da linha do tempo do projeto no desenvolvimento de software. [CMMI]

Indicador de desempenho de teste *Test performance indicator* Métrica de alto nível de eficácia e/ou eficiência utilizada para guiar e controlar o desenvolvimento progressivo de teste (por exemplo, porcentagem de detecção de defeito - PDD).

Indicador-chave de desempenho *Key performance indicator* Ver indicador de desempenho.

Infraestrutura de teste *Test infrastructure* Artefatos organizacionais necessários para realizar os testes. Eles consistem em ambientes de teste, ferramentas de teste, ambiente de escritório e procedimentos.

Inicialização (IDEAL) *Initiating (IDEAL)* Fase dentro do modelo IDEAL, onde o terreno está previsto para um esforço de melhoria bem sucedido. A fase inicial consiste nas atividades: estimulo para a mudança, estabelecimento do contexto, construção do patrocínio e estabelecer a infraestrutura. *Ver também IDEAL.*

Inspeção *Inspection* Revisão realizada pelos pares para detectar defeitos e baseada no exame visual de documentos, por exemplo, violações dos padrões de desenvolvimento e não conformidade da documentação de nível mais alto. Trata-se da técnica de revisão mais formal e, portanto, está sempre baseada em um procedimento documentado. [posterior a IEEE 610 e a IEEE 1028]. *Ver também revisão por pares.*

Inspetor *Inspector* Ver revisor.

Instabilidade *Installability* Capacidade que um produto de software tem para ser instalado em um ambiente específico [ISO 9126]. *Ver também portabilidade.*

Instrumentação *Instrumentation* Inserção de um código adicional no programa a fim de coletar informações sobre o comportamento do programa durante sua execução (por exemplo, para medir a cobertura de código).

Instrumentador *Instrumenter* Ferramenta de software utilizada para proceder com a instrumentação.

Instrumentador de programa *Program instrumenter* Ver instrumentador.

Integração *Integration* Processo de combinação de componentes ou sistemas em conjuntos maiores.

Integração funcional *Functional integration* Abordagem de integração que combina os componentes ou sistemas com a finalidade de fazer com que a funcionalidade básica funcione o mais rápido possível. *Ver também teste de integração.*

Inteligência emocional *Emotional intelligence* Habilidade, capacidade e competência para identificar, avaliar e gerenciar as emoções próprias, de outros e de grupos.

Interoperabilidade *Interoperability* Capacidade do produto de software de interagir com um ou mais componentes especificados ou sistemas. [posterior a ISO 9126] *Ver também funcionalidade.*

Item de cobertura *Coverage item* Entidade ou propriedade utilizada como base para a cobertura de teste (por exemplo, classes de equivalência ou sentenças de código).

Item de configuração *Configuration item* Agregação de hardware, software ou ambos, que é modelada para gerenciamento de configuração e tratado como uma entidade única do processo de gerenciamento de configuração. [IEEE 610]

Item de teste *Test item* Elemento individual a ser testado. Normalmente, há um objeto de teste e vários itens de teste. *Ver também objeto de teste.*

L

LCSAJ *LCSAJ* Cobertura de Sequência de Código Linear e Salto (Linear Code Sequence And Jump). Consiste nos três itens seguintes (convencionalmente identificados por número de linhas em uma lista de código fonte): início da sequência linear em sentenças executáveis, fim da sequência linear e linha alvo para a qual o fluxo de controle é transferido ao final da sequência linear.

Lider de inspeção *Inspection leader* Ver moderador.

Lider de teste *Test leader* Ver gerente de teste.

Linguagem script *Scripting language* Linguagem de programação na qual os scripts de testes executáveis são escritos e utilizados por uma ferramenta de execução de testes (por exemplo, ferramenta de captura/recuperação).

M

Manifesto ágil *Agile manifesto* Declaração de valores que fundamentam o desenvolvimento ágil de software. Estes valores são: » Os indivíduos e suas interações sobre procedimentos e ferramentas. » Software funcionando sobre documentação abrangente. » Colaboração do cliente sobre a negociação do contrato. » Responder à mudança sobre o seguimento de um plano.

Manifesto de melhoria do processo de teste *Test process improvement manifesto* Declaração que ecoa o manifesto ágil, e define os valores para a melhoria do processo de teste. Os valores são: » Flexibilidade sobre o detalhamento de processos. » Melhores práticas sobre modelos. » Orientação de implantação sobre orientação do

---

processo. » Revisão em pares sobre a garantia de qualidade (departamentos). » Orientação do negócio sobre a orientação do modelo.

Mantenabilidade *Maintainability* Facilidade com que um produto de software pode ser modificado para corrigir defeitos, atender a novos requisitos, facilitar manutenções futuras ou adaptar-se a um ambiente alterado. [ISO 9126]

Manutenção *Maintenance* Modificação de um produto de software após a implantação para corrigir defeitos, melhorar o desempenho ou outros atributos, ou adaptar o produto a um ambiente modificado. [IEEE 1219]

Mapa mental *Mind-map* Diagrama usado para representar palavras, idéias, tarefas ou outros itens ligados e organizados em torno de uma palavra chave ou idéia central. Mapas Mentais são utilizados para gerar, visualizar, estruturar e classificar idéias e como um auxílio no estudo, organização, resolução de problemas, tomada de decisão e da escrita.

Máquina de estado finito *Finite state machine* Modelo computacional que consiste em um número finito de estados e de transições entre esses estados, possivelmente com ações de acompanhamento. [IEEE 610]

Marco *Milestone* Determinado ponto de um projeto no qual os “entregáveis” definidos (intermediários) e os resultados devem estar prontos.

Marcos da qualidade *Quality gate* Um marco especial em um projeto. Estão localizados entre as fases de um projeto fortemente dependente do resultado de uma fase anterior. Um marco de qualidade inclui a verificação formal dos documentos da fase anterior.

Mascaramento de defeito *Defect masking* Ocorrência na qual um defeito evita a detecção de outros. [posterior a IEEE 610]

Mascaramento de falha *Fault masking* Ver mascaramento de defeito.

Maturidade *Maturity* (1) Capacidade de uma organização com relação à eficácia e eficiência de seus processos e práticas de trabalho.

Ver também CMM (Capability Maturity Model), TMM (Modelo de Maturidade de Teste). (2) Capacidade do produto de software para evitar a falha como resultado de defeitos no software. [ISO 9126]

Ver também confiabilidade.

Medição *Measurement* Processo de atribuição de um número ou categoria a uma entidade para descrever um atributo da entidade. [ISO 14598]

Medida *Measure* Número ou categoria assinalada a um atributo de uma entidade através de uma medição. [ISO 14598]

Melhoria de processos *Process improvement* Programa de atividades destinadas a melhorar o desempenho e a maturidade dos processos da organização, e o resultado do programa. [CMMI]

Melhoria do Processo de Teste (TPI) *Test Process Improvement (TPI)* Estrutura contínua para a melhoria do processo de teste que descreve os principais elementos de um processo de teste eficaz. Especialmente voltada para o teste de sistema e para o teste de aceitação.

Melhoria no processo de software *Software process improvement* Programa de atividades concebido para melhorar o desempenho e a maturidade dos processos da organização e os resultados desse programa. [posterior a CMMI]

Método de classificação por árvore *Classification tree method* Técnica de modelagem de teste caixa-preta onde os casos de teste, descritos por uma árvore de classificação, são modelados para executar combinações de domínios de entrada e/ou de saída. [Grochtmann]

Métrica *Metric* Escala de medição e o método utilizado para a medição. [ISO 14598]

Métrica de cobertura de Chow *Chow's coverage metrics* Ver cobertura N-switch.

Mitigação de risco *Risk mitigation* Ver controle de risco.

Modelagem de gráfico de causa-efeito *Cause-effect graphing* Técnica de modelagem de testes caixa-preta na qual os casos de testes são modelados a partir de gráficos de causa-efeito. [BS 7925/2]

Modelagem de teste *Test design* (1) Ver especificação do projeto de teste. (2) Processo de transformar os objetivos gerais do teste em condições de teste tangíveis e casos de teste.

Modelo baseado em conteúdo *Content-based model* Modelo de processo que fornece uma descrição detalhada de boas práticas de engenharia, por exemplo, práticas de teste.

Modelo de ciclo de vida *Lifecycle model* Particionamento da vida de um produto ou projeto em fases. [CMMI] *Ver também o ciclo de vida do software.*

Modelo de crescimento da confiabilidade *Reliability growth model* Modelo que mostra o crescimento em termos de confiabilidade ao longo do tempo, após testes contínuos em um componente ou sistema, como resultado da eliminação dos defeitos que resultam em falhas de confiabilidade.

Modelo de desenvolvimento incremental *Incremental development model* Ciclo de desenvolvimento, onde um projeto é dividido em uma série de incrementos, cada um dos quais fornece uma parte da funcionalidade dos requisitos de projeto geral. Os requisitos são priorizados e entregues por ordem de prioridade no incremento adequado. Em algumas (mas não todas) as versões do modelo de ciclo de vida, cada subprojeto segue um "mini-modelo V" com seu próprio desenho, codificação e fases de teste.

Modelo de desenvolvimento interativo *Iterative development model* Ciclo de vida de desenvolvimento no qual um projeto é normalmente dividido em iterações. Dá-se o nome de interação a um laço (loop) de desenvolvimento

---

completo que resulta em uma liberação (interna ou externa) de um produto executável, num subconjunto do produto final em desenvolvimento, o qual cresce de interação em interação até tornar-se o produto final.

Modelo de excelência da EFQM (European Foundation for Quality Management) *EFQM (European Foundation for Quality Management) excellence model*
Framework não prescritivo para o sistema de gerenciamento da qualidade da organização, definido pela European Foundation for Quality Management (EQFM), baseado nos cinco critérios de "habilitação" (cobrindo o que uma organização faz) e quatro critérios de "resultados" (cobrindo o que uma organização alcança).

Modelo de maturidade *Maturity model* Coleção estruturada de elementos que descrevem certos aspectos da maturidade de uma organização auxiliando na definição e compreensão dos processos. Um modelo de maturidade geralmente fornece uma linguagem comum, visão compartilhada e um quadro de priorização de ações de melhoria.

Modelo de processo *Process model* Framework onde os processos da mesma natureza são classificados em um modelo global (por exemplo, um modelo de melhoria de teste).

Modelo V *V-model* Estrutura que descreve as atividades do ciclo de vida do desenvolvimento de um software, desde a especificação de requisitos até a manutenção. O modelo V ilustra como as atividades de teste podem ser integradas em cada fase do ciclo de vida do desenvolvimento de um software.

Moderador *Moderator* É o líder ou o principal responsável pela inspeção ou por outro processo de revisão.

Modificabilidade *Changeability* Capacidade que um produto de software tem em permitir que modificações específicas sejam implementadas. [ISO 9126] *Ver também manutenibilidade.*

Modo de falha *Failure mode* Manifestação física ou funcional de uma falha. Por exemplo, um sistema em modo de falha pode ser caracterizado pelo funcionamento lento, saídas incorretas, ou a paralização completa da execução. [IEEE 610]

Módulo *Module* Ver componente.

Monitor *Monitor* Ferramenta de software ou dispositivo de hardware que é executado de forma concorrente junto ao componente ou sistema sob teste, que supervisiona, registra e/ou analisa o comportamento do componente ou sistema. [posterior a IEEE 610]

Monitoramento de teste *Test monitoring* Tarefa do gerenciamento de testes que lida com as atividades relacionadas às verificações periódicas da condição de um projeto de teste. São preparados relatórios para comparar os resultados reais e os planejados. *Ver também gerenciamento de teste.*

MPS *Spi* Ver melhoria no processo de software.

MTBF *Mtbf* Ver tempo médio entre falhas.

MTTR *Mttr* Ver tempo médio de reparo.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

N

Não conformidade *Non-conformity* Trata-se do não atendimento a requisito especificado. [ISO 9000]

Nível de maturidade *Maturity level* Grau de melhoria de processo através de um conjunto predefinido de áreas de processo na qual todos os objetivos são alcançados. [TMMI]

Nível de risco *Risk level* Importância de um risco, tal como definida pelas suas características de impacto e probabilidade. O nível de risco pode ser usado para determinar a intensidade do teste a ser realizado. O nível de risco pode ser expresso de forma qualitativa ou quantitativa.

Nivel de teste *Test level* Grupo de atividades de teste organizadas e gerenciadas conjuntamente. Um nível de teste está ligado às responsabilidades do projeto. Podemos citar como exemplos teste de componente, teste de integração, teste de sistema e teste de aceitação. [posterior a TMap]

Nota de lançamento *Release note* Documento que identifica os itens de um teste, suas configurações, seu estado atual e outras informações de entrega fornecidas pelo desenvolvimento para serem testados, e, possivelmente aos outros stakeholders, no início da fase de execução do teste. [posterior a IEEE 829]

Número ciclomático *Cyclomatic number* Ver complexidade ciclomática.

O

Objetivo de teste *Test objective* Razão ou finalidade por trás da modelagem e da execução de um teste.

Objeto de teste *Test object* Componente ou sistema a ser testado. *Ver também item de teste.*

Operabilidade *Operability* Capacidade do produto de software em habilitar o usuário a operá-lo e controlá-lo. [ISO 9126] *Ver também a usabilidade.*

Oráculo *Oracle* Ver oráculo de teste.

Oráculo de teste *Test oracle* Fonte utilizada para determinar os resultados esperados e compará-los com os resultados reais produzidos pelo software em teste. Um oráculo pode ser um sistema existente (para um *benchmark*), um manual de usuário ou o conhecimento especializado de um indivíduo, porém, não deve ser o código. [posterior a Adrion]

P

Pacote diário *Daily build* Atividade de desenvolvimento na qual um sistema completo é compilado e liberado diariamente (normalmente durante a noite), de forma que um sistema consistente esteja sempre disponível de acordo com as últimas atualizações.

Padrão *Standard* Conjunto formal de requisitos, eventualmente obrigatório, desenvolvido e usado para prescrever formas coerentes de trabalho ou para fornecer orientações (por exemplo, normas ISO/IEC, normas IEEE, e as normas de organização). [posterior a CMMI]

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Painel de controle *Dashboard* Representação de medições dinâmicas de desempenho operacional para algumas organizações ou atividades, usando métricas representadas por metáforas visuais, como "marcadores", "contadores" e outros dispositivos semelhantes às do painel de um automóvel, de modo que os efeitos de eventos ou atividades podem ser facilmente entendidos e relacionados com os objetivos operacionais. *Ver também painel de controle corporativo, scorecard.*

Par definição-utilização *Definition-use pair* Associação da definição com a utilização de uma variável. As variáveis podem ser usadas para computar uma informação (por exemplo, multiplicação), ou para direcionar a execução de um caminho (uso “predicado”).

Partição de equivalência *Equivalence partition* Parte de uma entrada ou saída de domínio para o qual o comportamento de um componente ou sistema é assumido ser o mesmo, baseado na especificação.

Particionamento de equivalência *Equivalence partitioning* Técnica de modelagem de teste caixa-preta na qual os casos de testes são modelados para executar a partir de partições de equivalência. Em principio, os casos de teste são modelados para cobrir cada partição pelo menos uma vez.

Percentual de detecção de defeitos (PDD) *Defect detection percentage (DDP)* Número de defeitos encontrados em uma fase de teste, dividido pelo número encontrado em todas as fases do teste ou em qualquer outro meio depois.

Percentual de detecção de falha *Fault detection percentage (FDP)* Ver percentual na detecção de defeitos (PDD)

Perfil de carga *Load profile* Especificação da atividade que um componente ou sistema a ser testado pode ter na produção. Um perfil de carga consiste em um determinado número de usuários virtuais que transformam um conjunto definido de operações em um período de tempo especificado e de acordo com um perfil pré-operacional. *Ver também o perfil de operação.*

Perfil de desempenho *Performance profiling* Definição de perfis de usuário no desempenho, carga e/ou testes de estresse. Os perfis devem refletir o uso antecipado ou real de um componente ou sistema com base em um perfil operacional, portanto, com carga de trabalho esperada. *Ver também o perfil de carga, perfil operacional.*

Perfil operacional *Operational profile* Representação de um conjunto distinto de tarefas executadas por um componente ou sistema, possivelmente com base no comportamento do usuário ao interagir com o componente ou sistema, e suas probabilidades de ocorrência. Uma tarefa é tanto lógica quanto física e pode ser executada por várias máquinas ou executada em segmentos de tempo não-contíguos.

Planejamento de teste *Test planning* Atividade de criação ou atualização de um plano de teste.

Plano de melhoria de teste *Test improvement plan* Plano para atingir os objetivos de melhoria no processo de teste organizacional baseado no entendimento completo dos atuais pontos fortes e fracos dos processos de teste da organização e seus ativos. [posterior a CMMI]

Plano de teste *Test plan* Documento descrevendo o escopo, abordagem, recursos e cronograma das atividades de teste que se destina. Ela identifica, entre outros, itens de teste, recursos a serem testados, tarefas de teste, executor de cada tarefa, grau de independência do testador, o ambiente de teste, as técnicas de projeto de teste e

---

critérios de entrada e de saída a serem usados, as razões de sua escolha, e os eventuais riscos que exigem planos de contingência. É um registro do processo de planejamento de teste. [posterior a IEEE 829]

Plano de teste de fase *Phase test plan* Plano de teste que normalmente aborda uma fase de teste. *Ver também plano de teste.*

Plano de teste de nível *Level test plan* Plano de teste que aborda um nível de teste. *Ver também plano de teste.*

Plano de teste do projeto *Project test plan* Ver plano mestre de teste.

Plano mestre de teste *Master test plan* Plano de teste que aborda múltiplos níveis de teste. *Ver também plano de teste.*

Política de teste *Test policy* Documento de alto nível que descreve os princípios, a abordagem e os principais objetivos da organização de um teste.

Ponteiro *Pointer* Item de dado que especifica o local de outro item de dado, por exemplo, um item de dados que especifica o endereço do registro do próximo funcionário a ser processado. [IEEE 610]

Ponteiro perdido *Wild pointer* Ponteiro que referencia a existência ou não de um local que está fora do escopo. *Ver também ponteiro.*

Ponto de entrada *Entry point* Sentença executável ou etapa do processo que define um ponto em que um determinado processo destina-se a começar.

Ponto de saída *Exit point* Uma sentença executável ou passo de processo que define um ponto no qual um dado processo deve terminar.

Portabilidade *Portability* Facilidade com que o produto de software pode ser transferido de um ambiente de hardware ou software para outro. [ISO 9126]

Pós-condição *Postcondition* Condições de ambiente e de estado que devem ser atendidas após a execução de um teste ou de um procedimento de teste. Precondição *Precondition* Condições de ambiente e de estado que devem ser atendidas antes que um componente ou sistema possa ser executado com um determinado teste ou procedimento de teste.

Pré-teste *Pretest* Ver teste de entrada.

Prioridade *Priority* Nível de importância (do negócio) designado a um item (por exemplo, defeito).

Problema *Problem* Ver defeito.

Procedimento de teste *Procedure testing* Teste destinado a assegurar que o componente ou sistema pode funcionar em conjunto com novos ou existentes procedimentos de negócios de usuários ou procedimentos operacionais.

Processo *Process* Conjunto de atividades inter-relacionadas que transformam entradas em saídas. [ISO 12207]

Processo de teste *Test process* O processo de teste compreende fundamentalmente o planejamento e controle, modelagem e análise, implementação e execução, registro e critério de saída, e as atividades de fechamento.

Processos Críticos de Teste (CTP) *Critical Testing Processes (CTP)* Modelo baseado em conteúdo para a melhoria do processo de teste, construída em torno de doze processos críticos. Estes incluem processos altamente visíveis de missão crítica em que o desempenho afeta os lucros e reputação da empresa.

Programação em pares *Pair programming* Abordagem de desenvolvimento de software pela qual as linhas de código (produção e/ou teste) de um componente são escritas por dois programadores trabalhando em um único computador. Implicitamente, isto significa que revisões de código em tempo real são realizadas.

Projeto *Project* Conjunto único de atividades coordenadas e controladas com datas de início e fim, comprometidas a atingir um objetivo conforme requisitos específicos, incluindo as limitações de tempo, custo e recursos. [ISO 9000]

Pseudo-aleatório *Pseudo-random* Série que parece ser aleatória, mas que, de fato, foi gerada de acordo com alguma sequência pré-estabelecida.

Q

Qualidade *Quality* Grau até o qual um componente, sistema ou processo atende aos requisitos especificados e/ou às necessidades e expectativas do usuário/consumidor. [posterior a IEEE 610]

Qualidade baseada na construção *Manufacturing-based quality* Visão de qualidade, na qual é determinada pelo grau em que um produto ou serviço está em conformidade com os objetivos e exigências do projeto. A qualidade decorre do processo(s) utilizado. [posterior a Garvin] *Ver também a qualidade baseada no produto, qualidade baseada na transcendência, qualidade baseada no usuário, qualidade baseada no valor.*

Qualidade baseada na transcendência *Transcendent-based quality* Visão de qualidade onde ela não pode ser definida com precisão, mas a reconhecemos quando a vemos ou estamos cientes de sua ausência quando ela não estiver contemplada. A qualidade depende da percepção e os sentimentos afetivos de um indivíduo ou grupo de indivíduos para um produto. [posterior a Garvin] *Ver também qualidade baseada na fabricação, qualidade baseada no produto, qualidade baseada no usuário, qualidade baseada no valor.*

Qualidade baseada no produto *Product-based quality* Visão de qualidade, onde é baseada em um conjunto bem definido de atributos de qualidade. Esses atributos devem ser medidos de forma objetiva e quantitativa. As diferenças na qualidade dos produtos do mesmo tipo podem ser rastreadas até a forma como os atributos de qualidade específicos têm sido implementados. [posterior a Garvin] *Ver também qualidade baseada na fabricação, qualidade baseada em atributos, qualidade baseada na transcendencia, qualidade baseada no usuário, a qualidade baseada no valor.*

Qualidade baseada no usuário *User-based quality* Visão de qualidade, onde a qualidade é a capacidade de satisfazer as necessidades e desejos do usuário. Um produto ou serviço que não satisfaça estas necessidades é improvável que tenha quaisquer usuários. Este é um contexto de abordagem, condicionada à qualidade, já que as características de diferentes empresas requerem diferentes qualidades de um produto. [posterior a Garvin] *Ver também qualidade baseada na fabricação, qualidade baseada no produdo, a qualidade baseada na transcendência, qualidade baseada no valor.*

Qualidade baseada no valor *Value-based quality* Visão da qualidade, onde ela é definida pelo preço. Um produto ou serviço de qualidade é aquele que fornece o desempenho desejado, a um custo aceitável. A qualidade é determinada por meio de um processo de decisão com os interessados sobre os trade-offs entre tempo, esforço e aspectos de custo. [posterior a Garvin] *Ver também a qualidade baseada na produção, qualidade baseada no produto, qualidade baseada no usuário.*

Qualidade de software *Software quality* Totalidade das funcionalidades e características de um produto de software que se baseia na sua habilidade de satisfazer as necessidades declaradas ou implícitas. [posterior a ISO 9126]

Qualificação *Qualification* Processo de demonstrar a capacidade de cumprir os requisitos especificados. Observe que o termo "qualificado" é usado para designar uma situação correspondente. [ISO 9000]

R

Rastreabilidade *Traceability* Habilidade de identificar itens relacionados em documentos e softwares (por exemplo, requisitos e testes associados). *Ver também rastreabilidade horizontal e rastreabilidade vertical.*

Rastreabilidade horizontal *Horizontal traceability* Rastreamento dos requisitos para um nível de teste por meio de camadas de documentação de testes (por exemplo, plano de teste e especificação de modelagem, caso de teste, procedimento ou para roteiro de teste).

Rastreabilidade vertical *Vertical traceability* Rastreamento de requisitos por meio de camadas de documentação de desenvolvimento dos componentes.

Rational Unified Process *Rational Unified Process*
Processo proprietário de desenvolvimento de software que consiste quatro fases do ciclo de vida de projeto: incepção, elaboração, construção e transição.

Recuperabilidade *Recoverability* Capacidade de um produto de software para estabelecer novamente um nível específico de desempenho e de recuperar os dados diretamente afetados em caso de falha. [ISO 9126]. *Ver também confiabilidade.*

Redator *Scribe* Nome dado à pessoa que registra cada defeito mencionado ou sugestão dada para a melhoria do processo durante uma reunião de revisão, em um formulário de registro. O redator deve assegurar que o formulário de registro possa ser lido e entendido.

Registrador *Recorder* Ver redator.

Registrar incidente *Incident logging* Gravação dos detalhes de qualquer incidente ocorrido, por exemplo, durante o teste.

Registrar teste *Test logging* Processo pelo qual as informações sobre o teste executado são registradas em um registro de teste.

Registro da execução do teste *Test run log* Ver registro de teste.

Registro de teste *Test log* Registro cronológico das informações relevantes sobre a execução dos testes. [IEEE 829]

Relato da situação *Status accounting* Elemento do gerenciamento de configuração que consiste na gravação e relato das informações necessárias para gerenciar a configuração eficazmente. Estas informações incluem uma lista da identificação de configuração aprovada, a condição das alterações de configuração propostas e o estado de implementação das alterações aprovadas. [IEEE 610]

Relatório de avaliação *Assessment report* Documento que resume as avaliações de resultados, como por exemplo, conclusões, recomendações e pontos relevantes. *Ver também processo de avaliação de teste.*

Relatório de avaliação de teste *Test evaluation report* Documento produzido ao final do processo de teste e que resume todas as atividades de teste e seus resultados. Contém também uma avaliação do processo de teste e as lições aprendidas no referido processo.

Relatório de bug *Bug report* Ver relatório do defeito.

Relatório de defeito *Defect report* Documento que relata qualquer falha em um componente ou sistema que possa fazer com este componente ou sistema deixe de desempenhar sua função requisitada. [posterior a IEEE 829]

Relatório de desvio *Deviation report* Ver relatório de incidente.

Relatório de incidente *Incident Report* Documento que relata qualquer evento ocorrido durante o teste, que requer uma investigação. [posterior a IEEE 829]

Relatório de incidente de teste *Test incident report* Ver relatório de incidente.

Relatório de incidente de teste de software *Software test incident report* Ver relatório de incidente.

Relatório de problema *Problem report* Ver relatório de defeito.

Relatório de progresso de teste *Test progress report* Documento que resume as atividades de teste e os resultados produzidos em intervalos regulares, para relatar o progresso das atividades de teste contra uma linha de base (como o plano de teste inicial) e para comunicar os riscos e as alternativas que exigem uma decisão de gestão.

Relatório de resumo de teste *Test summary report* Documento que resume as atividades e os resultados de um teste. Contém também uma avaliação dos itens de teste correspondentes versus os critérios de saída. [posterior a IEEE 829]

Relatório de teste *Test report* Ver relatório do resumo de teste e relatório de progresso de teste.

Relatório de transmissão de item *Item transmittal report* Ver nota de liberação.

Reportar incidente *Incident report* Documento que notifica a ocorrência de qualquer evento, (por exemplo, durante o teste) que requer investigação. [posterior o IEEE 829]

Representação contínua *Continuous representation* Estrutura do CMM onde os níveis de capacidade provêm uma ordem recomendada para a abordagem de melhoria de processos dentro das áreas de processo especificado. [CMMI]

Representação por estágios *Staged representation* Estrutura de modelo em que a satisfação das metas de um conjunto de áreas de processo estabelece um nível de maturidade. Cada nível constrói uma base para os níveis subsequentes. [posterior a CMMI]

Reproducibilidade de teste *Test reproducibility* Atributo de um teste que indica se os mesmos resultados são produzidos a cada vez que o teste é executado.

Reprovação de teste *Test fail* Ver falhar.

Requisito *Requirement* Condição ou capacidade necessária a um usuário para resolver um problema ou atingir um objetivo que deve ser atendido ou presente em um componente ou sistema para satisfazer um contrato, padrão, especificação ou outro documento formal imposto. [posterior a IEEE 610]

Requisito de teste *Test requirement* Ver condição de teste.

Requisito funcional *Functional requirement* Especifica determinada função que um componente ou sistema deve desempenhar. [IEEE 610]

Requisito não funcional *Non-functional requirement* Requisito que não diz respeito à funcionalidade, mas a atributos como confiabilidade, eficiência, usabilidade, manutenibilidade e portabilidade.

Requisitos testáveis *Testable requirements* Grau até o qual um requisito se declara em termos que permitam o estabelecimento da modelagem de teste (e subsequentemente de casos de teste) e a execução de testes para determinar se os requisitos foram atendidos. [posterior a IEEE 610]

Resultado *Result* Consequência ou resultado da execução de um teste. Incluem saídas para as telas, alterações de dados, envio de relatórios e mensagens. *Ver também resultado real e resultado esperado.*

Resultado de decisão *Decision outcome* Resultado de uma decisão (a qual, por sua vez, determina os desvios a serem seguidos).

Resultado de teste *Test result* Ver resultado.

Resultado esperado *Expected result* Comportamento previsto pela especificação, ou por outra fonte, de um componente ou sistema sob determinadas condições.

Resultado real *Actual result* Comportamento produzido/observado quando um componente ou sistema é testado

Reteste *Re-testing* Teste que executa casos de teste reprovados durante sua última execução. Este procedimento é realizado para verificar o sucesso das ações corretivas.

Retrospectiva de reunião *Retrospective meeting* Reunião no final de um projeto durante o qual os membros da equipe avaliam o projeto e aprendem as lições que podem ser aplicadas para o próximo projeto.

Retrospectiva do projeto *Project retrospective* Maneira estruturada para captar as lições aprendidas e criar planos de ação específicos para melhorar no próximo projeto ou fase seguinte do projeto.

Reunião pós projeto *Post-project meeting* Ver restrospectiva de reunião

Revisão *Review* Avaliação das condições de um produto ou projeto para averiguar discrepâncias em relação aos resultados planejados e para recomendar melhorias. Como exemplos de revisão, podemos citar as revisões de gerenciamento, as revisões informais, revisões técnicas, as inspeções e os acompanhamentos. [posterior a IEEE 1028]

Revisão ad hoc *Ad hoc review* Ver revisão informal.

Revisão de gerenciamento *Management review* Avaliação sistemática da aquisição de software, do fornecimento, do desenvolvimento, da operação ou do processo de manutenção. Tal avaliação pode ser feita pelo gerenciamento, ou em seu nome, a fim de monitorar o progresso, determinar o estado dos planos e dos cronogramas, confirmar os requisitos e seus sistemas de alocação ou de avaliar a eficácia da abordagem de gerenciamento para fins de otimização. [posterior a IEEE 610 e IEEE 1028]

Revisão de testabilidade *Testability review* Verificação detalhada das bases de um teste a fim de determinar se a base de teste está num nível adequado de qualidade para agir como documento de entrada do processo de teste. [posterior a TMap]

Revisão formal *Formal review* Revisão caracterizada por procedimentos e requisitos documentados, por exemplo, inspeção.

Revisão informal *Informal review* Revisão que não é baseada em um procedimento formal (documentado).

Revisão por pares *Peer review* Revisão do trabalho de um produto de software feita por colegas do produtor do produto, com a finalidade de identificar defeitos e apontar melhorias. Como exemplo, podemos citar a inspeção, a revisão técnica e o acompanhamento.

Revisão técnica *Technical review* Discussão realizada entre pares buscando o consenso sobre o tipo de abordagem técnica a ser utilizada. [Gilb e Graham, IEEE 1028]. *Ver também revisão por pares.*

Revisor *Reviewer* Pessoa envolvida no processo de revisão e que identifica e descreve as anomalias encontradas no produto ou projeto sendo revisto. Os revisores podem ser escolhidos para representar diferentes pontos de vista e papéis neste processo.

Risco *Risk* Fator que pode resultar em consequências futuras negativas; normalmente expresso em termos de impacto e possibilidade.

Risco de produto *Product risk* Risco diretamente relacionado ao objeto do teste. *Ver também risco.*

Risco de projeto *Project risk* Riscos relacionados com a gestão e controle do projeto de teste (por exemplo, falta de pessoal, prazos rigorosos, mudança de requisitos). *Ver também risco.*

Robustez *Robustness* Grau até o qual um componente ou sistema pode funcionar corretamente na presença de entradas inválidas ou de condições ambientais estressantes. [IEEE 610]. *Ver também tolerância ao erro e tolerância à falta.*

RUP *RUP*
Ver Rational Unified Process

S

Saída *Output* Uma variável (seja armazenada dentro ou fora de um componente) que é escrita por um componente.

Saída esperada *Expected outcome* Ver resultado esperado.

Scorecard *Scorecard* Representação resumida de medições de desempenho que representam o progresso na implementação dos objetivos de longo prazo. Um scorecard fornece medições estáticas do desempenho durante ou no final de um intervalo definido.
Ver também balanced scorecard, dashboard.

Script de teste *Test script*
Termo normalmente utilizado para se referir a uma especificação de procedimento de teste, especialmente em testes automatizados.

Scrum *Scrum* Estrutura iterativa e incremental para o gerenciamento de projetos comumente usado com o desenvolvimento ágil de software. *Ver também desenvolvimento ágil de software.*

Segurança *Safety* Capacidade que um produto de software tem para alcançar níveis de risco aceitáveis tanto para pessoas como negócios, software, propriedade ou meio ambiente dentro de um contexto de utilização específico. [ISO 9126]

Semeamento de erro *Error seeding* Ver semeamento de falha.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Semeamento de falhas *Fault seeding* Processo de adição intencional de defeitos conhecidos que já estão no componente ou sistema para efeito de controle da taxa de detecção e remoção, e estimar o número de defeitos restantes. IEEE [610]

Sensibilização de caminho *Path sensitizing* Escolha de um conjunto de valores de entrada para forçar a execução de um dado caminho.

Sentença *Statement* Entidade em uma linguagem de programação, que normalmente é a menor unidade indivisível de execução.

Sentença executável *Executable statement* Sentença que, quando compilada, é traduzida em código objeto, e que poderá ser executada através de procedimentos quando um programa está sendo executado, podendo também executar uma ação em dados.

Sentença fonte *Source statement* Ver sentença.

Sessão de teste *Test session* Período ininterrupto de tempo gasto na execução dos testes. Em testes exploratórios, cada sessão de teste está focada em uma carta, mas os testadores podem também explorar novas oportunidades ou problemas durante a sessão. O testador cria e executa os casos de teste em tempo real e de seus registros de progresso. *Ver também testes exploratórios.*

Severidade *Severity* Grau de impacto que um defeito tem sobre o desenvolvimento ou operação de um componente ou sistema. [posterior a IEEE 610]

Simulação *Simulation* Representação de características comportamentais selecionadas de um sistema físico ou abstrato por outro sistema. [ISO 2382/1]

Simulador *Simulator* Dispositivo, programa de computador ou sistema utilizado durante o teste e que se comporta ou opera como um dado sistema quando recebe um conjunto de entradas controladas. [posterior a IEEE 610 e a DO178b]. *Ver também emulador.*

Sistema *System* Coleção de componentes organizados para realizar uma função específica ou conjunto de funções. [IEEE 610]

Sistema de segurança critica *Safety critical system* Sistema cuja falha ou mau funcionamento pode resultar em morte ou lesões graves a pessoas, perda ou danos graves ao equipamento, ou dano ambiental.

Sistema de sistemas *System of systems* Vários sistemas heterogêneos e distribuídos que estão inseridos em redes em vários níveis e em múltiplos domínios interligados, para tratar problemas e objetivos de grande escala e inter-disciplinares, geralmente sem uma estrutura de gestão comum.

Situação de teste *Test situation* Ver condição de teste.

Software *Software* Programas de computador, procedimentos e possível documentação associada e dados relativos à operação de um sistema de computador. IEEE [610]

Software comercial de prateleira *Commercial off-the-shelf software* Ver software de prateleira.

Software customizado *Bespoke software* Software desenvolvido especificamente para um conjunto de usuários ou clientes. Seu oposto é o software de prateleira.

Software de prateleira *Off-the-shelf software* Produto de software que é desenvolvido para o mercado em geral, ou seja, para um grande número de clientes, e que é entregue para muitos clientes em formato idêntico.

Software Failure Mode and Effect Analysis (SFMEA) *Software Failure Mode and Effect Analysis (SFMEA)*
Ver Failure Mode and Effect Analysis (FMEA).

Software Failure Mode, Effects, and Criticality Analysis (SFMECA) *Software Failure Mode, Effects, and Criticality Analysis (SFMECA)*
Ver Failure Mode, Effects, and Criticality Analysis (FMECA).

Software Fault Tree Analysis (SFTA) *Software fault tree analysis (SFTA)*
Ver Fault Tree Analysis (FTA).

Software padrão *Standard software*
Ver software de prateleira.

Software Usability Measurement Inventory (SUMI) *Software Usability Measurement Inventory (SUMI)*
Uma técnica de teste baseada em questionário para medição da qualidade de software a partir do ponto de vista do usuário final. [Veenendaal]

STEP *STEP*
Ver Systematic Test and Evaluation Process (STEP).

Subcaminho *Subpath* Sequência de sentenças executáveis dentro de um componente.

Substitutibilidade *Replaceability* Capacidade que um produto de software tem para ser utilizado no lugar de outro produto de software específico para os mesmos fins e no mesmo ambiente. [ISO 9126]. *Ver também portabilidade.*

Suite de caso de teste *Test case suite* Ver suite de teste.

Suite de teste *Test suite* Conjunto de vários casos de teste para um componente ou sistema sendo testado, no qual a pós-condição de um teste é frequentemente utilizada como precondição para o próximo.

Suposição de erro *Error guessing* Técnica de modelagem de teste, onde a experiência do testador é usada para antecipar defeitos que podem estar presentes no componente ou sistema em teste, como resultado de erros cometidos, e para modelar testes especificamente para expô-las.

Systematic Test and Evaluation Process (STEP) *Systematic Test and Evaluation Process (STEP)*
Metodologia estruturada de testes, também usado como um modelo baseado em conteúdo para melhorar o processo de teste. Sistemática de teste e avaliação de processo (STEP) não exige que as melhorias ocorram em uma ordem específica. *Ver também modelo baseado no conteúdo.*

T

Tabela de decisão *Decision table* Tabela que mostra as combinações de entradas e/ou estímulos (causas) com suas saídas e/ou ações (efeitos) associadas, que podem ser utilizadas para projetar casos de testes.

Tabela de decisão de causa-efeito *Cause-effect decision table* Ver tabela de decisão.

Tabela de estado *State table* Grade mostrando as transições resultantes em cada estado, combinado com cada evento possível, mostrando ambas as transições válidas e inválidas.

Taxa de falha *Failure rate* Razão do número de falhas de uma dada categoria para uma dada unidade de medida, por exemplo, falhas por unidade de tempo, número de transações, número de execução de computadores. [IEEE 610]

Taxonomia do bug *Bug taxonomy* Ver taxonomia do defeito.

Taxonomia do defeito *Defect taxonomy* Um sistema (hierárquico) de categorias descrito para auxiliar na classificação de defeitos reproduzidos.

Técnica baseada em defeitos *Defect based technique* Ver técnica de modelagem de testes baseada em defeitos.

Tecnica baseada em especificação *Specification-based technique* Ver tecnica de modelagem de teste caixa-preta.

Técnica baseada na experiência *Experience-based technique* Ver técnica de modelagem de teste baseada na experiência.

Técnica de caixa-branca *White-box technique* Ver técnica de modelagem de teste caixa-branca.

Técnica de caixa-preta *Black box technique* Ver técnica de modelagem de teste caixa-preta.

Técnica de especificação de teste *Test specification technique* Ver técnica de modelagem de teste.

Técnica de execução de teste *Test execution technique* Método utilizado para desempenhar a real execução do teste, seja manual ou automaticamente.

Técnica de modelagem de caso de teste *Test case design technique* Ver técnica de modelagem de teste.

Técnica de modelagem de teste *Test design technique* Procedimento utilizado para derivar e/ou selecionar casos de teste.

JUCEMAR FORMIGONI CÂNDIDO (238879@profe.sed.sc.gov.br)

Técnica de modelagem de teste baseada em defeitos *Defect based test design technique* Procedimento para obter e/ou selecionar casos de teste para uma ou mais categorias de defeitos, com testes desenvolvidos a partir do que é conhecido sobre uma categoria específica de defeito. *Ver também taxonomia de defeitos.*

Técnica de modelagem de teste baseada na experiência *Experience-based test design technique* Procedimento para obter e/ou selecionar os casos de teste baseados na experiência, conhecimento e intuição do testador.

Técnica de modelagem de teste baseado na estrutura *Structure-based test design technique* Ver técnica de modelagem de teste caixa-branca.

Técnica de modelagem de teste baseado na especificação *Specification-based test design technique* Ver tecnica de modelagem de teste caixa-preta.

Técnica de modelagem de teste caixa-preta. *Black box test design technique* Técnica de derivar e/ou selecionar casos de teste considerando as especificações, funcionais ou não-funcionais, de um componente ou sistema, sem referenciar sua estrutura interna.

Técnica de modelagem de teste caixa-branca *White-box test design technique* Procedimento para derivar e/ou selecionar casos de teste baseado em uma análise da estrutura interna de um componente ou sistema.

Técnica de modelagem de teste estrutural *Structural test design technique* Ver técnica de modelagem de teste caixa-branca.

Técnica de modelagem de teste funcional *Functional test design technique* Procedimento que deriva e/ou seleciona casos de testes com base em uma análise da especificação da funcionalidade de um componente ou sistema sem fazer referência à sua estrutura interna. *Ver também técnica de modelagem de teste caixa-preta.*

Técnica de modelagem de teste não funcional *Non-functional test design technique* Procedimento que deriva e/ou seleciona os casos de teste para teste não funcional. Baseia-se na análise da especificação de um componente ou sistema sem referir-se à sua estrutura interna. *Ver também técnica de modelagem de teste caixa-preta.*

Técnica de teste *Test technique* Ver técnica de modelagem de teste.

Tempo médio de reparo *Mean time to repair* Média aritmética (média) do tempo que um sistema levará para se recuperar de eventuais falhas. Isso normalmente inclui testes para garantir que o defeito foi resolvido.

Tempo médio entre falhas *Mean time between failures*
A média aritmética (média) de tempo entre falhas de um sistema. O MTBF é normalmente parte de um modelo de crescimento da confiabilidade que supõe que o sistema é imediatamente reparado como parte de um processo de correção de defeitos. *Ver também o modelo de crescimento da confiabilidade.*

Test Maturity Model (TMM) *Test Maturity Model (TMM)*
Estrutura de cinco níveis utilizada para a melhoria do processo de teste relaciona-se ao Modelo de Maturidade de Capacidade (CMM) que descreve os principais elementos de um processo eficaz de teste.

Test Maturity Model Integrated (TMMi) *Test Maturity Model Integrated (TMMI)*
Framework estagiado em cinco níveis para a melhoria do processo de teste, relacionadas com o Capability Maturity Model Integration (CMMI), descreve os principais elementos de um processo de teste eficaz.

Testabilidade *Testability* Capacidade do produto de software de permitir que o software, quando modificado, seja validado. [ISO 9126] *Ver também mantenabilidade.*

Testador *Tester* Profissional habilitado e envolvido no teste de um componente ou sistema.

Testar *Testing* Processo que consiste em todas as atividades do ciclo de vida, tanto estáticas quanto dinâmicas, voltadas para o planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados a fim de determinar se elas satisfazem os requisitos especificados e demonstrar que estão aptas para sua finalidade e para a detecção de defeitos.

Teste *Test* Conjunto de um ou mais casos de teste [IEEE 829]

Teste ad hoc *Ad hoc testing*
Teste realizado informalmente sem a preparação ou utilização de técnicas de modelagem reconhecidas, e sem definição prévia dos resultados esperados.

Teste ágil *Agile testing* Prática de teste para um projeto que utiliza metodologias ágeis, como Extreme Programming (XP), que trata o processo de desenvolvimento como o cliente de teste e enfatiza o paradigma "test-first design". *Ver também desenvolvimento orientado ao teste.*

Teste aleatório *Random testing* Técnica de modelagem de teste caixa-preta na qual os casos de teste são selecionados, possivelmente por meio de um algoritmo de geração pseudo-aleatória para atender um perfil operacional. Esta técnica pode ser utilizada para testar atributos não funcionais, tais como confiabilidade e desempenho.

Teste alfa *Alpha testing* Teste operacional, simulado ou real, realizado por usuários/clientes potenciais ou por uma equipe independente de testes no ambiente dos desenvolvedores, mas fora da organização desenvolvedora da solução. O Teste Alfa é frequentemente realizado para sistemas de prateleira ("software off-the-shelf") como forma de teste de aceite interna.

Teste alternado *Pairwise testing* Teste caixa-preta de um projeto no qual os casos de teste são projetados para executar todas as possíveis combinações discretas de cada par de parâmetros de entrada. *Ver também teste de matriz ortogonal.*

Teste baseado em checklist *Checklist-based testing*
Técnica de modelagem de teste baseada na experiência, pelo qual o testador utiliza uma lista de alto nível de itens a serem observados, verificados, ou lembrados, ou um conjunto de regras ou critérios que um produto deverá ser verificado. *Ver também teste baseado na experiência.*

Teste baseado em código *Code-based testing* Ver teste caixa-branca. Teste baseado em especificação *Specification-based testing* Ver teste caixa-preta.

Teste baseado em modelagem *Design-based testing* Abordagem de testes em que os casos de teste são modelados baseando-se na arquitetura e/ou modelagens detalhadas de um componente ou sistema (por exemplo, testes de interfaces entre componentes ou sistemas).

Teste baseado em processos de negócios *Business process-based testing* Abordagem na qual os casos de teste são modelados com base em descrições e/ou no conhecimento dos processos dos negócios.

Teste baseado em requisito *Requirements-based testing* Abordagem de teste na qual os casos de testes são modelados com base nos objetivos e nas condições de teste derivados dos requisitos, por exemplo, testes que exercitam funções específicas ou investigam atributos não funcionais, tais como confiabilidade ou usabilidade.

Teste baseado em risco *Risk-based testing* Abordagem de testes para reduzir o nível de riscos de produtos e informar as partes interessadas do seu estado, a partir dos estágios iniciais de um projeto. Ela envolve a identificação dos riscos do produto e do uso dos níveis de risco para orientar o processo de teste.

Teste baseado em sessão *Session-based testing* Abordagem de testes em que as atividades de teste são planejadas como sessões ininterruptas de concepção e execução do teste, muitas vezes utilizada em conjunto com testes exploratórios.

Teste baseado na estrutura *Structure-based testing* Ver teste caixa-branca.

Teste básico *Smoke test* Subconjunto de todos os casos de testes definidos/planejados que cobre a principal funcionalidade de um componente ou sistema, para averiguar as principais funções de um programa em funcionamento sem se preocupar com maiores detalhes. A realização diária de testes de construção e teste básico está entre as melhores práticas do ramo. *Ver também teste de entrada.*

Teste beta *Beta testing* Teste operacional realizado por usuários/consumidores existentes/potenciais em um local externo, sem envolvimento dos desenvolvedores, a fim de determinar se um componente ou sistema satisfaz, ou não, as necessidades de usuários/consumidores e se encaixa dentro dos processos dos negócios. O teste beta é frequentemente utilizado como uma forma de teste de aceitação externa para softwares de prateleira (off-the-shelf software), possibilitando avaliar o feedback do mercado.

Teste big-bang *Big-bang testing*
Tipo de teste integrado no qual os elementos de hardware, software, ou ambos, são combinados e testados de uma única vez, ao invéz de serem testados em estágios. [posterior a IEEE 610]. *Ver também teste integrado.*

Teste bottom-up *Bottom-up testing* Abordagem incremental do teste de integração, na qual os componentes de níveis mais baixo são testados em primeiro lugar, e, então utilizados para facilitar o teste de componentes de níveis mais alto. Este processo é repetido até que o componente no topo da hierarquia seja testado. *Ver também teste de integração.*

Teste caixa branca *White-box testing* Teste baseado na análise da estrutura interna de um componente ou sistema.

Teste caixa-clara *Clear-box testing* Ver teste caixa-branca.

Teste caixa-preta *Black box testing* Execução de teste, funcional ou não funcional, sem levar em consideração a estrutura interna do componente ou sistema.

Teste com scripts *Scripted testing* Execução do teste seguindo previamente uma sequência documentada de testes.

Teste completo *Complete testing* Ver teste exaustivo.

Teste de aceitação do usuário *User acceptance testing* Ver teste de aceite.

Teste de aceite *Acceptance testing* Teste formal relacionado às necessidades dos usuários, requisitos e processos de negócios. É realizado para estabelecer se um sistema satisfaz ou não os critérios de aceitação e para possibilitar aos usuários, aos clientes e às outras entidades autorizadas decidir aceitar ou não determinado sistema. [posterior a IEEE 610]

Teste de aceite de produção *Production acceptance testing* Ver teste de aceite operacional.

Teste de aceite de site *Site acceptance testing* Teste de aceitação realizado por usuários/consumidores em seu próprio local a fim de determinar se um componente ou sistema satisfaz ou não as necessidades dos usuários/consumidores e se enquadra dentro dos processos de negócios, normalmente incluindo tanto hardware como software.

Teste de aceite operacional *Operational acceptance testing* Testes operacionais na fase de teste de aceitação, geralmente realizado em um ambiente (simulado) operacional em operações e/ou pessoal de administração de sistemas com foco em aspectos operacionais, por exemplo, recuperabilidade, comportamento dos recursos, instalabilidade e conformidade técnica. *Ver também o teste operacional.*

Teste de acessibilidade *Accessibility testing* Teste que determina a facilidade com a qual usuários portadores de deficiências possam utilizar determinado componente ou sistema. [Gerrard]

Teste de acurácia *Accuracy testing* Processo de teste com o objetivo de determinar a acurácia de um produto de software.

Teste de adequação *Suitability testing* Processo de testes para determinar a adequação de um produto de software.

Teste de algoritmo *Algorithm test* [TMap] Ver teste de desvio.

Teste de arco *Arc testing* Ver teste de desvio.

Teste de armazenamento *Storage testing* Ver teste de utilização de recurso.

Teste de arranjo ortogonal *Orthogonal array testing* Forma sistemática de testar todas as combinações de pares de variáveis usando matrizes ortogonais. Reduz significativamente o número de todas as combinações de variáveis para testar todas as combinações de pares. *Ver também teste em pares.*

Teste de benchmark *Benchmark test* (1) Padrão de referência para realização de medições e comparações. (2) Teste utilizado para comparar componentes ou sistemas entre si, ou em relação a um padrão, conforme visto em (1). [posterior a IEEE 610].

Teste de caixa de vidro *Glass box testing* Ver teste caixa-branca.

Teste de caminho *Path testing* Técnica de modelagem de teste caixa-branca na qual os casos de teste são modelados para executar caminhos.

Teste de carga *Load testing* Tipo de teste de desempenho realizado para avaliar o comportamento de um componente ou sistema com carga crescente, por exemplo, número de usuários paralelo e/ou o número de transações, para determinar qual a carga pode ser manipulada por um componente ou sistema. *Ver também teste de desempenho, teste de estresse.*

Teste de caso de uso *Use case testing* Técnica de modelagem de teste de caixa-preta na qual os casos de teste são modelados para executar cenários de usuário.

Teste de cenário *Scenario testing* Ver teste de caso de uso.

Teste de cenários de usuário *User scenario testing* Ver teste de caso de uso.

Teste de ciclo de processo *Process cycle test* Técnica de modelagem de teste caixa-preta na qual os casos de teste são modelados para executar procedimentos e processos comerciais. [TMap]

Teste de cobertura lógica *Logic-coverage testing* Ver teste caixa-branca. [Myers]

Teste de combinação de condição *Condition combination testing* Ver teste de condição múltipla.

Teste de combinação de condição de desvio *Branch condition combination testing* Ver teste de condição múltipla.

Teste de comparação *Back-to-back testing* Teste em que duas ou mais variantes de um componente ou sistema são executados com as mesmas entradas, tendo as saídas comparadas, e analisadas em casos de discrepâncias. [IEE 610].

Teste de comparação elementar *Elementary comparison testing* Técnica de modelagem de teste caixa-preta nas quais os casos de testes são modelados para executar combinações de entradas utilizando o conceito de cobertura de determinação de condição. [TMap]

Teste de compatibilidade *Compatibility testing* Ver testes de interoperabilidade.

Teste de componente *Component testing* Teste individual de componente de software. [posterior a IEEE 610]

Teste de concorrência *Concurrency testing* Teste realizado para determinar como a ocorrência de duas ou mais atividades dentro de um mesmo intervalo de tempo, alcançada por entrelaçamento das atividades ou por execução simultânea, é tratada pelo componente ou sistema. [posterior a IEEE 610]

Teste de condição *Condition testing* Técnica para modelagem de testes caixa-branca onde os casos de teste são modelados para executar o resultado de uma condição.

Teste de condição de decisão *Decision condition testing* Técnica de modelagem de teste caixa-branca na qual os casos de teste são modelados para executar resultados de condição e resultados de decisão.

Teste de condição determinada *Condition determination testing* Técnica para modelagem de testes em caixa-branca onde os casos de teste são modelados para executar o resultado de uma condição simples que afeta o resultado da decisão.

Teste de condição múltipla *Multiple condition testing* Técnica de modelagem de teste caixa-branca no qual os casos de teste são criados para executar combinações de resultados de condição simples (dentro de uma instrução).

Teste de condição múltipla modificada *Modified multiple condition testing* Ver teste de determinação de condição.

Teste de confiabilidade *Reliability testing* Processo que determina a confiabilidade de um produto de software.

Teste de confiança *Confidence test* Ver teste básico.

Teste de configuração *Configuration testing* Ver teste de portabilidade.

Teste de confirmação *Confirmation testing* Ver reteste.

Teste de conformidade *Compliance testing* Processo de testes para determinar a conformidade do componente ou sistema.

Teste de conversão *Conversion testing* Testes de software usados para converter dados de sistemas existentes para uso em sistemas substitutos.

Teste de decisão *Decision testing* Técnica de modelagem de testes caixa-branca na qual os casos de testes são projetados para executar os resultados de decisões.

Teste de decisão de condição modificada *Modified condition decision testing* Ver teste de determinação de condição.

Teste de desempenho *Performance testing* Processo que determina o desempenho de um produto de software. *Ver também teste de eficiência.*

Teste de desenvolvimento *Development testing* Teste formal ou informal conduzido durante a implementação de um componente ou sistema, normalmente realizado no ambiente de desenvolvimento pelos desenvolvedores. [posterior a IEEE 610]

Teste de desvio *Branch testing* Técnica de modelagem de teste caixa-branca na qual os casos de teste são modelados para executarem os desvios.

Teste de documentação *Documentation testing* Testa a qualidade da documentação, por exemplo, guia do usuário ou guia de instalação.

Teste de eficiência *Efficiency testing* Processo de teste para determinar a eficiência de um produto de software.

Teste de entrada *Intake test* Instância especial do teste básico que decide se o componente ou sistema está pronto para testes mais detalhados. Este teste normalmente é realizado no início da fase de execução de teste. *Ver também teste básico.*

Teste de escalabilidade *Scalability testing* Teste que determina a escalabilidade de um produto de software.

Teste de estado finito *Finite state testing* Ver teste de transição de estado.

Teste de estresse *Stress testing* Avalia um sistema ou componente em relação e além dos limites de seus requisitos especificados. [IEEE 610]. *Ver também teste de carga.*

Teste de fluxo de dados *Data flow testing* Técnica de modelagem de teste caixa-branca na qual casos de teste são modelados para executar definições e utilizar pares de variáveis.

Teste de funcionalidade *Functionality testing* Realizado para determinar a funcionalidade de um produto de software.

Teste de instabilidade *Installability testing* O processo de testar a instalabilidade de um produto de software. *Ver também teste de portabilidade.*

Teste de integração *Integration testing* Teste realizado com a finalidade de expor defeitos nas interfaces e nas interações entre componentes ou sistemas integrados. Ver também teste de integração de componente e teste de integração de sistema.

Teste de integração de componentes *Component integration testing* Testes realizados para expor os defeitos nas interfaces e interação entre os componentes integrados.

Teste de integração de sistema *System integration testing* Testar a integração de sistemas e pacotes e/ou as interfaces para outras organizações externas (por exemplo, Intercâmbio Eletrônico de Dados, Internet).

Teste de integração em larga escala *Integration testing in the large* Ver teste de integração de sistema.

Teste de integração em pequena escala *Integration testing in the small* Ver teste de integração de componente.

Teste de integridade de banco de dados *Database integrity testing* Testa os métodos e processos utilizados para acessar e gerenciar o banco de dados a fim de assegurar que os métodos de acesso, os processos e as regras dos dados funcionem conforme esperado e que, durante o acesso ao banco de dados, estes não sejam corrompidos, removidos inesperadamente, atualizados ou criados.

Teste de integridade de dados *Data integrity testing* Ver teste de integridade de banco de dados.

Teste de interface *Interface testing* Tipo de teste de integração que testa interfaces entre componentes ou sistemas.

Teste de interoperabilidade *Interoperability testing* Nome dado ao processo que determina a interoperabilidade de um produto de software. *Ver também teste de funcionalidade.*

Teste de isolamento *Isolation testing* Testa componentes individuais, isolando-os dos componentes do meio. Se houver necessidade, podem-se simular os componentes do meio com simuladores (stubs) e controladores (drivers).

Teste de LCSAJ *LCSAJ testing* Técnica de modelagem de teste caixa-branca na qual os casos de teste são modelados para executar LCSAJs.

Teste de link *Link testing* Ver teste de integração de componente.

Teste de lógica orientada *Logic-driven testing* Ver teste caixa-branca.

Teste de mantenabilidade *Maintainability testing* Processo que determina a mantenabilidade de um produto de software. Teste de manutenção *Maintenance testing* Testa as alterações feitas em um sistema operacional ou o impacto de um ambiente alterado em um sistema operacional.

Teste de mesa *Desk checking* Teste de um software ou uma especificação por meio da simulação manual de sua execução. *Ver também análise estática.*

Teste de migração *Migration testing* Ver teste de conversão.

Teste de módulo *Module testing* Ver teste de componente.

Teste de mutação *Mutation testing* Ver teste de comparação.

Teste de padrões *Standards testing* Ver teste de atendimento.

Teste de partição *Partition testing* Ver particionamento de equivalência. [Beizer]

Teste de perfil operacional *Operational profile testing* Teste estatístico que utiliza um modelo de operações de sistema (para tarefas de curta duração) e da probabilidade de uso mais comum. [Musa]

Teste de portabilidade *Portability testing* Processo que determina a portabilidade de um produto de software.

Teste de programa *Program testing* Ver teste de componente.

Teste de recuperabilidade *Recoverability testing* Processo que determina a recuperabilidade de um produto de software. *Ver também teste de confiabilidade.*

Teste de recuperação *Recovery testing* Ver teste de recuperabilidade.

Teste de regressão *Regression testing* Teste realizado em um programa previamente testado após alguma modificação feita e com a finalidade de assegurar que defeitos não tenham sido introduzidos ou mascarados nas áreas não alteradas do software como resultado da referida modificação. Este teste é realizado quando o software ou seu ambiente é alterado.

Teste de regulamentação *Regulation testing* Ver teste de atendimento.

Teste de robustez *Robustness testing* Teste que determina a robustez de um produto de software.

Teste de sanidade *Sanity test* Ver teste básico.

Teste de segurança *Safety testing* Teste que determina a segurança de um produto de software.

Teste de sentença *Statement testing* Técnica de modelagem de teste caixa-branca na qual os casos de teste são modelados para executar sentenças.

Teste de servicibilidade *Serviceability testing* Ver teste de mantenabilidade.

Teste de sintaxe *Syntax testing* Técnica de modelagem de teste caixa-preta na qual os casos de teste são modelados com base nas definições do domínio de entrada e/ou no domínio de saída.

Teste de sistema *System testing* Testa um sistema integrado para verificar se ele atende aos requisitos especificados. [Hetzel]

Teste de tabela de decisão *Decision table testing* Técnica de modelagem de teste caixa-preta na qual os casos de testes são projetados para executar as combinações de entradas e/ou estímulos (causas) exibidos em uma tabela de decisão [Veenendaal]. *Ver também tabela de decisão.*

Teste de transição de estados *State transition testing* Técnica de modelagem caixa-preta na qual os casos de testes são modelados para executar transições de estados válidas e inválidas. *Ver também teste N-switch.*

Teste de unidade *Unit testing* Ver teste de componente.

Teste de usabilidade *Usability testing* Teste que determina a extensão até a qual o produto de software é entendido, fácil de aprender, fácil de operar e atraente para os usuários sob condições especificas. [posterior a ISO 9126]

Teste de usuário *User test* Teste no qual os usuários da vida real se envolvem na avaliação da usabilidade de um componente ou sistema.

Teste de utilização de recurso *Resource utilization testing* Processo que determina a utilização de recursos por um dado produto de software. *Ver também teste de eficiência.*

Teste de valor limite *Boundary value testing* Ver análise de valor limite.

Teste de volume *Volume testing* Teste que submete o sistema a grandes volumes de dados. *Ver também teste de recurso e utilização.*

Teste dinâmico *Dynamic testing* Testes que envolvem a execução de um software, um componente ou um sistema.

Teste do macaco *Monkey testing* Teste realizado por meio de uma seleção aleatória de uma grande variedade de entradas e apertando botões ao acaso, ignorando como o produto está sendo usado.

Teste em campo *Field testing* Ver beta teste.

Teste em pares *Pair testing* Duas pessoas, por exemplo, dois testadores, um desenvolvedor e um testador, ou um usuário final e um testador, trabalham juntos para descobrir defeitos. De modo geral, eles compartilham o mesmo computador e alternam-se no controle durante o teste.

Teste em threads *Thread testing* Versão do teste de integração de componente na qual a integração progressiva de componentes segue a implementação de subconjuntos de requisitos, ao contrário da integração de componentes por níveis de hierarquia.

Teste estático *Static testing* Teste de um componente ou sistema em especificação ou implementação sem a execução do referido programa, por exemplo, revisões ou análise estática.

Teste estatístico *Statistical testing* Técnica de modelagem de teste no qual um modelo de distribuição estatística da entrada é usado para construir casos de teste representativos. *Ver também teste de perfil operacional.*

Teste estrutural *Structural testing* Ver teste caixa-branca.

Teste exaustivo *Exhaustive testing* Abordagem na qual a suite de teste abarca todas as combinações de valores e precondições de entrada.

Teste exploratório *Exploratory testing* Técnica de modelagem de teste informal na qual o testador controla ativamente a modelagem dos testes enquanto estes são executados, e utiliza a informação obtida durante o teste para modelar testes novos e melhorados. [posterior a Bach]

Teste funcional *Functional testing* Teste baseado em uma análise da especificação de funcionalidade de um componente ou sistema. *Ver também teste caixa-preta*

Teste incremental *Incremental testing* Teste no qual os componentes ou sistemas são integrados e testados sozinhos ou em pequenos grupos por vez, até que todos os componentes ou sistemas sejam integrados e testados.

Teste inválido *Invalid testing* Utiliza valores de entrada que devem ser rejeitados pelo componente ou sistema. *Ver também tolerância ao erro.*

Teste não funcional *Non-functional testing* Teste dos atributos de um componente ou sistema que não se relacionam com a funcionalidade (por exemplo, confiabilidade, eficiência, usabilidade, manutenibilidade e portabilidade).

Teste negativo *Negative testing* Visa mostrar que um componente ou sistema não funciona. O teste negativo se refere mais à atitude do testador do que a uma abordagem de teste específica ou uma técnica de modelagem de teste, por exemplo, o teste com valores de entrada inválidos ou com exceções. [posterior a Beizer].

Teste N-switch *N-switch testing* Forma de teste de transição de estado na qual os casos de teste são modelados para executar todas as sequências de N+1 válidas. [Chow]. *Ver também teste de transição de estado.*

Teste operacional *Operational testing* Realizado com a finalidade de avaliar um componente ou sistema em seu ambiente operacional. [IEEE 610]

Teste orientado a dados *Data driven testing* Técnica de script que armazena a entrada de teste e os resultados esperados em uma tabela ou planilha, de modo que um único script de controle pode executar todos os testes na tabela. O teste orientado a dados é frequentemente usado para dar suporte à aplicação de ferramentas de execução de teste, tais como ferramentas de captura/reprodução. [Fewster e Graham] *Ver também teste orientado a palavra-chave.*

Teste orientado a palavra de comando *Action word driven testing* Ver teste orientado a palavra chave.

Teste orientado a palavra-chave *Keyword driven testing* Técnica de script que utiliza arquivos de dados para conter não só dados de teste e os resultados esperados, mas também palavras-chave relacionadas com a aplicação que está sendo testada. As palavras-chave são interpretadas por scripts especiais de suporte que são chamadas pelo script de controle do teste. *Ver também teste orientado a dados.*

Teste sujo *Dirty testing* Ver teste negativo.

Teste top-down *Top-down testing*
Abordagem incremental para o teste de integração, onde o componente no topo da hierarquia do componente é testado em primeiro lugar, com componentes de baixo nível simulados por um simulador. Componentes testados

---

são usados para testar os componentes de nível inferior. O processo é repetido até que os componentes de nível mais baixo sejam testados. *Ver também teste de integração.*

Testware *Testware* Artefatos produzidos durante o processo de teste e requeridos para planejar, projetar e executar testes, entre eles documentação, roteiros, entradas, resultados esperados, procedimentos de preparação e de limpeza, arquivos, bancos de dados, ambiente e qualquer software adicional ou utilitários utilizados no teste. [posterior a Fewster e Graham] Testware de automação *Automated testware* Testware utilizado em testes automatizados, tais como os scripts de uma ferramenta. Tipo de risco *Risk type* Conjunto de riscos agrupados por um ou mais elementos comuns, tais como um atributo de qualidade, causa, localização, ou o efeito potencial de risco. Um conjunto específico de tipos de risco do produto está relacionado ao tipo de teste que pode mitigar (controle) qual o tipo de risco. Por exemplo, o risco de má interpretação das interações com o usuário pode ser atenuado por meio de testes de usabilidade. Tipo de teste *Test type* Grupo de atividades de teste que testa um componente ou sistema enfocando um objetivo de teste específico, ou seja, funcional, usabilidade, regressão, etc. Um tipo de teste pode acontecer em um ou mais níveis ou fases de teste. [posterior a TMap] Tolerância ao erro *Error tolerance* Habilidade de um sistema ou componente para continuar operando normalmente apesar da presença de entradas errôneas. [posterior a IEEE 610] Tolerância à falha *Fault tolerance* Capacidade que um produto de software tem para manter um nível específico de desempenho em casos de faltas (defeitos) de software ou de infração de sua interface específica. [ISO 9126]. *Ver também confiabilidade.* Total Quality Management (TQM) *Total Quality Management (TQM)* Abordagem de gestão de toda a organização centrada na qualidade, baseada na participação de todos os seus membros e visando o sucesso em longo prazo através da satisfação do cliente, e os benefícios para todos os membros da organização e para a sociedade. Consiste em planejamento, organização, direção, controle e garantia. [posterior a ISO 8402]
TPG *TPG*
Ver Test Process Group.
TQM *TQM*
Ver Total Quality Management (TQM).

Transição de estado *State transition* Transição entre dois estados de um componente ou sistema. Tratamento de exceção *Exception handling* Comportamento de um componente ou sistema em resposta a uma entrada incorreta de um usuário humano ou de outro componente ou sistema. Trilha de auditoria *Audit trail* Caminho pelo qual a entrada original para um processo (por exemplo, dados) pode ser rastreada através do processo, tendo a saída do processo como um ponto de partida. Isso facilita a análise de defeitos e permite que um processo de auditoria possa ser levado adiante. [posterior a TMap]

U

Unidade *Unit* Ver componente. Usabilidade *Usability* Capacidade que um software tem de ser entendido, aprendido, utilizado e atraente para o usuário quando utilizado sob determinadas condições. [ISO 9126] Utilização de recurso *Resource utilization* Capacidade do produto de software em usar quantidades e tipos adequados de recursos, por exemplo, a quantidade de memória principal e secundária usada pelo programa e os tamanhos dos arquivos temporários necessários ou em excesso, quando o software executa suas funções sob condições estabelecidas. [posterior a ISO 9126] *Ver também eficácia.*

V

Validação *Validation* Confirmação, por meio de exames e do fornecimento de evidências objetivas, que os requisitos de uso ou aplicação específica tencionada foram atendidos. [ISO 9000] Valor de entrada *Input value* Instância de uma entrada. *Ver também entrada.* Valor de saída *Output value* Instância de uma saída. *Ver também saída.* Valor limite *Boundary value* Valor de entrada ou de saída que está na fronteira de uma partição equivalente

---

ou na menor distância incremental em qualquer limite da fronteira (por exemplo, o menor ou maior valor de um intervalo).

Variável *Variable* Elemento de armazenamento em um computador que pode ser acessado por um programa de software referindo-se a ele por um nome. Vazamento de memória *Memory leak* Falha de acesso de memória devido a um defeito na lógica de alocação dinâmica de armazenamento de um programa, que faz com que o programa falhe ao liberar a memória após tê-la utilizado, eventualmente causando falha no programa e/ou nos processos concorrentes em função da falta de memória. Verificação *Verification* Confirmação por meio de exame e do fornecimento de evidências objetivas que os requisitos especificados foram atendidos. [ISO 9000]
Verificador *Checker*
Ver revisor.

W

WBS *WBS*
Ver Work Breakdown Structure.
Wide band delphi *Wide band delphi*
Técnica de estimativa de teste baseada em especialização que visa produzir uma estimativa precisa utilizando a sabedoria coletiva dos membros da equipe.
Work Breakdown Structure (WBS) *Work breakdown structure*
Arranjo de elementos de trabalho e seu relacionamento com outros elementos ou a um produto final. [CMMI]
