NOME DO PROJETO		
PADRÕES APLICADOS AO PROJETO **PADRÕES DE PROJETO**		
Preparado por	Gustavo Costa; Leticia Raquel; Vitor Costa; Willian Gontijo	Versão: 1.0
Aprovado por:	Eduardo Habbib	17/09/2015

Teste de Aceitação do Usuário

Já no final da criação do sistema se encontra o Teste de Aceitação do Usuário. Esse teste tem por finalidade, examinar as funcionalidades do software no seu estágio final. Apesar do nome, o teste continua sendo realizado pela equipe desenvolvedora. A explicação para o nome é a forma como ele é estruturado:

> "(...) começa no fim do teste de integração, quando componentes individuais já foram exercitados, o software está completamente montado como um pacote, e os erros de interface foram descobertos e corrigidos. (...)". O teste focaliza ações visíveis ao usuário e saídas do sistema reconhecidas pelo usuário. (Pressman).[i]
>
> É um 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. (Glossário , ISTQB). [ii]

É de responsabilidade do cliente ou do usuário do sistema; os interessados (stakeholders) também podem ser envolvidos. . (Syllabus , ISTQB).[iii]

Esse teste se caracteriza por uma fase de "caixa-preta". Nele, são testadas funcionalidades do sistema através de entradas de dados de vários tipos. Esses dados não necessariamente se relacionam com a estrutura do

sistema. Assim, é realizado no produto, testes de “força bruta”, isso, porque as variações de dados colocados nas entradas são de todas as ordens. Quanto mais variáveis representando grupos, tipo de dados, mais rica é a análise realizada no teste. As vantagens de utilização estão na simulação de um ambiente real, onde o software estará sujeito a inserções de informações e requisições no sistema de maneiras diversas, podendo, através de defeitos no código, apresentar um erro no programa e, por fim, gerar uma falha para o usuário.

Formas de implementar um Teste de Usuário são:

- **Teste de Aceitação de usuário**

Averigua aceitação por um usuário com perfil do negócio.

- **Teste Operacional de Aceite**

Um teste de aceitação pelo administrador do sistema. Esse irá testar suas principais funcionalidades, como Back-up, updates, etc.

- **Teste de aceitação de contrato e regulamento**

Atesta as coerências das exigências do software encomendado com base nas cláusulas de especificações do contrato. Verifica se o produto corresponde à encomenda. Averiguações como a concordância quanto ás normas governamentais são conferidas nessa etapa.

- **Alfa e Beta Teste (ou teste no campo)**

É um teste realizado, geralmente, na própria empresa desenvolvedora. Nela, uma equipe encarregada pelos testes coloca em prática as funcionalidades do sistema. Outra alternativa é levar até o produto, uma parcela de clientes que irá consumir o software para realizar testes.

• Teste Beta (ou teste no campo)

Aqui, os clientes testam o produto antes de ser lançado no mercado. É uma forma viável de obter um feedback dos consumidores.

Teste de integração

Sistemas de informação, sejam, web ou desktop, em quase todas as vezes, seguem padrões de desenvolvimento que separam o sistema como um todo em partes menores. Isso torna a implementação profissional e organizada. Na Engenharia de Software, Teste de Integração é o conjunto de averiguações sobre o sistema no momento de união de seus diversos módulos, esses, desenvolvidos e testados separadamente em etapas prévias.

A principal função do Teste de Integração é responder à uma pergunta recorrente durante a implementação de um sistema: Quando integrarmos tudo, o sistema irá funcionar? Assim, essa etapa situa-se após os testes nos módulos em separados, e antes da aplicação no sistema completo.

O teste de Integração segue uma sequência de especificações estabelecidas previamente no plano de teste, e resulta num sistema integrado e preparado para o Teste de Sistema - Teste do Sistema é um teste geral no sistema após este estar montado. O plano de Testes é um documento descrito no padrão IEEE 829 (Padrão 829 para Documentação de Teste de Software) que especifica o formato de documentos de testes de software e hardware, mas sem especificar como os testes devem ser reproduzidos. Entretanto, o Plano de Teste é geralmente constituído de fluxogramas detalhados de funcionamento durante os processos.

É possível realizar os Testes de Integração de diversas maeiras, mas as principais são, segundo o site da Microsoft[iv]:

- A abordagem top-down para testes de integração requer os módulos de nível mais alto de teste à serem integrados em primeiro lugar. Isto permite que a lógica de alto nível e o fluxo de dados a ser testada no início do processo e tende a minimizar a necessidade de drivers. No entanto, a necessidade de stubs – rotina fantasma pra teste- complica utilitários de gerenciamento e testes de baixo nível são realizados relativamente tarde no ciclo de desenvolvimento. Outra desvantagem dos testes de integração top-down é o seu fraco suporte para a liberação antecipada de funcionalidade limitada.

- A abordagem bottom-up requer as unidades de nível mais baixo à serem testadas e integradas em primeiro lugar. Estas unidades são freqüentemente referidas como módulos de serviços públicos. Ao utilizar esta abordagem, os módulos de utilidade são testados no início do processo de desenvolvimento e a necessidade de topos é minimizado. A desvantagem, porém, é que a necessidade de drivers complica o gerenciamento de testes e de alto nível lógica e dados de fluxo são testados tarde. Como a abordagem top-down, a abordagem bottom-up também fornece suporte precário para a liberação antecipada de funcionalidade limitada.

- A terceira abordagem, algumas vezes referida como a abordagem de guarda-chuva, requer testes dos dados funcionais e o fluxo de controle de caminhos. Em primeiro lugar, as entradas para as funções são integrados no padrão de baixo para cima discutido acima. As saídas de cada função são então integrados na forma de cima para baixo. A principal vantagem dessa abordagem é o grau de apoio para a liberação antecipada de funcionalidade limitada. Ele também ajuda a minimizar a necessidade de stubs e drivers. Os potenciais pontos fracos desta abordagem são significativos, no entanto, na medida em que pode ser menos sistemática do que as outras duas abordagens, conduzindo à necessidade de mais testes de regressão.

Teste de Regressão

Durante o processo de desenvolvimento ou manutenção de um sistema são comuns alterações em componentes já implementados. Desta forma, erros podem eventualmente surgir com a reintegração dos componentes modificados ao sistema como um todo. Quando um teste afirma um erro nesse contexto, diz-se que houve regresso. Assim, surge o Teste de Regressão. Em síntese, a função desse teste é: Garantir que o programa ainda satisfaz seus requisitos [Rothermel, 94].[v]

Uma definição retirada do glossário de termos do ISTQB [vi]- International Software Testing Qualification Board - diz o seguinte:

> 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.

Frequentemente, equipes responsáveis pela produção e cuidados com softwares não realizam Testes de Regressão, ou se realizam, não os fazem com devido rigor. Isso, muitas vezes, visando a rapidez na entrega do produto ao cliente. Entretanto, a implantação desse tipo de teste pode reduzir os riscos de problemas com o produto, e eventualmente, com o cliente.

Alguns tópicos importantes na realiz:

- **Rastreabilidade**

Manter um sistema mapeado e documentado facilita no momento de prever uma possível falha na alteração e reintegração de uma parte do produto.

- **Rotinas críticas**

Essas rotinas são, em suma, as quais não podem apresentar problemas. São normalmente as rotinas mais utilizados pelo programa. É necessário conhecer e tomar cuidados com essas rotinas.

- **Testes redundantes**

Um cuidado importante à ser tomado na implantação de testes no sistema é quanto à redundância nos mesmos. Um teste anteriormente programado e desatualizado em relação às novas exigências de detecção de falha será inútil para avanços na averiguação do produto.

- **Testes que falharam**

Levar em consideração teste anteriores que apresentaram falhar, assim como relatos dos clientes. Com isso, criar uma base de testes com abordagem dessas falhas nas futuras implementações dos Testes de Regressão. Assim, estruturando um teste mais amplo, eficiente e preciso.

- **Automatizar os testes de regressão**

Se o cliente procura produtos que sejam bons e que possam ficar prontos o mais rápido possível, o mais recomendável à uma equipe de desenvolvimento, em relação aos seus testes, é programa-los para execução automática. Assim, substituindo o teste manual, a versão automatizada economiza tempo dos desenvolvedores.

Algumas ferramentas que executam Testes de Regressão são:

- Rational functional tester - IBM
- mercury quick teste professional - HP
- JUnit - Java
- NUnit - .NET

Teste de Unidade

Um **teste de unidade** é aquele que testa uma única unidade do sistema. Ele a testa de maneira isolada, geralmente simulando as prováveis dependências que aquela unidade tem. E geralmente, em termos de códigos orientados a objeto (OO), estas unidades de trabalho são métodos de uma classe. O objetivo do teste unitário é assegurar que cada unidade está funcionando de acordo com sua especificação funcional. Estes tipos de testes são frequentemente escritos por desenvolvedores quando trabalham no código, para assegurar que a função específica está executando como esperado. Uma função deve ter muitos testes para dar cobertura a todos os caminhos possíveis do seu código. É importante resaltar que sozinho, o teste unitário não

testa o sistema como um todo, pórem ele é usado para assegurar que os blocos constituintes do software trabalham independentes dos demais.

Testes Unitários – Quando fazer? Porque fazer? O que testar?

Sempre é bom no início de cada ciclo de desenvolvimento do software testar de forma isolada cada complemento. Além disso, a cada construção do sistema, pois é mais fácil corrigir um problema pontual do que no futuro um problema complexo. Segue abaixo algumas das melhorias que o teste de unidade possibilita no desenvolvimento de um software.

1. Previne contra o aparecimento de “bugs” oriundos de códigos mal escritos.
2. Código testado é mais confiável.
3. Permite alterações sem medo (coragem)
4. Testa situações de sucesso e de falha.
5. Resulta em outras práticas XP como: Código coletivo, refatoração, integração contínua.
6. Serve como métrica do projeto (teste == requisitos)
7. Gera e preserva um “conhecimento” sobre as regras de negócios do projeto.

Sempre nós ficamos em dúvida sobre o que devemos testar em nossas classes, existem alguns macetes que podem nos ajudar a descobrir quais e quantos testes deverão ser escritos:

1. A principal regra para saber o que testar é: “Tenha criatividade para imaginar as possibilidades de testes”.
2. Comece pelas mais simples e deixe os testes “complexos“ para o final.
3. Use apenas dados suficientes (não teste 10 condições se três forem suficientes)
4. Não teste métodos triviais, tipo get e set.
5. No caso de um método set, só faça o teste caso haja validação de dados.

6. Achou um bug? Não conserte sem antes escrever um teste que o pegue (se você não o fizer, ele volta)!

Testes Unitários – JUnit

Existe algumas ferramentas que facilitam o processo te aplicação dos teste unitários algumas delas são: JUnit, TesteNG, PHPUnit, NUnit são exemplo; abaixo iremos especificar sobre o JUnit.

O JUnit é um framework que facilita o desenvolvimento e execução de testes unitários em código Java. O framework permite a criação de testes unitários. Além disso, está disponível como plug-in para os mais diversos IDE'S como Eclipse, Netbeans.

Os principais motivos que favorecem o uso desse framework são:

· JUnit pode verificar se cada unidade de código funciona da forma esperada.

· Facilita a criação, execução automática de testes e a apresentação dos resultados.

· É Orientado a Objeto.

· É gratuito.

Teste de Usabilidade

Teste de usabilidade é uma técnica de pesquisa utilizada para avaliar um produto ou serviço. Os testes são realizados com usuários que representam o público-alvo. Cada participante tenta realizar tarefas típicas enquanto o analista observa, ouve e anota.

Além disso, possui como objetivo verificar a aceitação por parte dos usuários usando como parâmetro a facilidade encontrada em utilizar o produto e detectar a origem de comportamentos inesperados/indesejados.

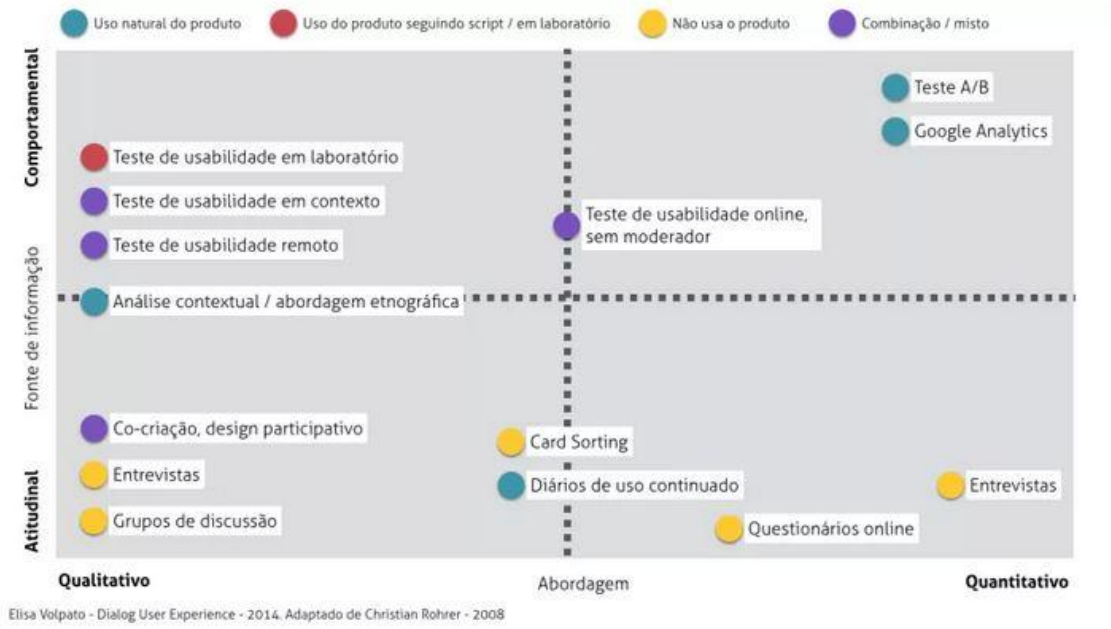
Há duas maneiras de se realizar a observação do comportamento do usuário quando se utiliza o produto. A primeira mais dispendiosa e que produz um resultado mais verídico com uso natural do produto e mais rico. Consiste em utilizar um laboratório de usabilidade.

Este laboratório é composto por duas salas ligadas por um espelho translucido e uma câmera de vídeo gravando o rosto e as atitudes e a interação do participante com o produto. A maneira mais econômica não utiliza desde laboratório e utiliza apenas câmera de vídeo e uma sala privada.

Além do mais, existem duas diferentes formas de se abordar este teste, a primeira consiste no eixo horizontal que se ramifica em: qualitativo e quantitativo. Na abordagem qualitativa o objetivo é identificar diferentes comportamentos, opiniões e atitudes sobre o produto. Já na abordagem quantitativa é medir quantas pessoas acham isso ou fazem aquilo e quantificar comportamentos mais comuns.

Já o eixo vertical se preocupa com a distinção do que o usuário diz sobre o produto sobre o que ele realmente faz sobre o produto. É de extrema importância para o avaliador saber identificar na opinião do participante o que é mais próximo da vivência deste.

O diagrama abaixo foi criado para o curso de teste de usabilidade, com base no artigo de Christian Rohrer (2008), com as práticas mais comuns em nosso mercado. Em resumo, a parte de cima do diagrama é focado em como as pessoas utilizam seu produto e a parte de baixo as impressões e opiniões que as pessoas tem sobre seu produto.



Em resumo: se você quer saber como as pessoas utilizam o seu produto, busque as técnicas do lado de cima do gráfico. Se quer saber a opinião que as pessoas têm sobre o seu produto, busque o lado de baixo.

Teste de Interface

O teste de interface é apenas uma entre várias formas de observar e medir a experiência do usuário. Em linhas gerais, a área de Interação Humano-Computador (IHC) investiga o projeto (design), avaliação e implementação de sistemas computacionais interativos para uso humano, juntamente com os fenômenos associados a esse uso.

Os estudos relacionados ao projeto de IHC se referem a como construir interfaces com alta qualidade. Para isto, são definidos métodos, modelos e diretrizes. Os estudos relacionados à avaliação de IHC, por sua vez, buscam avaliar a qualidade de um projeto de interface, tanto ao longo do processo de desenvolvimento como quando o software está pronto.

O teste de interface busca detectar no participante sua opinião e aceitação em relação a interface do produto com o usuário. Interface é o nome dado a toda a porção de um sistema com a qual um usuário mantém contato

ao utilizá-lo, tanto ativa quanto passivamente. A interface engloba tanto software quanto hardware (dispositivos de entrada e saída, tais como: teclados, mouse, tablets, monitores, impressoras e etc.).

O teste busca analisar em um participante do público alvo se a interface com o usuário do produto em questão gera um maior dinamismo nas ações ou apenas produz dificuldades. Como produto do teste também é coletado a opinião do participante para averiguar possíveis melhores e adequações a realidade e vivência deste possível participante do público alvo.

Bibliografia

Pressman, R. S. (2006). Engenharia de Software. Mcgraw-Hill, 6 a edição edição.

http://www.istqb.org/downloads/glossary.html

Syllabus Fundation, BSTQB, versão 2011br. Disponível em http://www.bstqb.org.br Glossário de termos, BSTQB, versão 2.1.1br. Disponível em http://www.bstqb.org.br

Prates, Raquel Oliveira, and Simone Diniz Junqueira Barbosa. "Avaliação de Interfaces de Usuário–Conceitos e Métodos." Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo. Vol. 6. 2003.

https://msdn.microsoft.com/en-us/library/aa292128%28VS.71%29.aspx

Rothermel , G.; Harrold, M. J.; " A Framework for Evaluating Regresseion Test Selection Techniques ". Proc. of the 16 th Int'l. Conf. on Softw. Eng., Sorrento, Italy, may/1994, p. 201-210.

http://www.istqb.org/downloads/glossary.html

http://www.istqb.org/downloads/glossary.html

JUnit - Implementando testes unitários em Java – Parte I http://www.devmedia.com.br/junit-implementando-testes-unitarios-em-java-parte-i/1432#ixzz3itdId2HZ

TDD: fundamentos do desenvolvimento orientado a testes http://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151#ixzz3itkbUmql