UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

DESENVOLVIMENTO DE UMA METODOLOGIA PARA TESTES EXPLORATÓRIOS

TRABALHO DE GRADUAÇÃO

Aluna: Diana Rúbia Rodrigues Ricardo (drrr@cin.ufpe.br)

Recife, 22 de Agosto de 2007.

UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

DESENVOLVIMENTO DE UMA METODOLOGIA PARA TESTES EXPLORATÓRIOS

TRABALHO DE GRADUAÇÃO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Departamento de Sistemas de Computação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Alexandre Marcos Lins de Vasconcelos (amlv@cin.ufpe.br)

Recife, 22 de Agosto de 2007.

Assinaturas

Este Trabalho de Graduação é resultado dos esforços da aluna Diana Rúbia Rodrigues Ricardo, sob a orientação do professor Alexandre Marcos Lins de Vasconcelos. Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste Trabalho de Graduação.

_____

Diana Rúbia Rodrigues Ricardo

_____

Alexandre Marcos Lins de Vasconcelos

Resumo

No mercado atual torna-se cada vez mais necessário um software com um alto nível de qualidade e que satisfaça o cliente. Para que os produtos sejam entregues com mais qualidade é vital a realização de testes de software.

Há empresas que não têm condições de implantar um processo de teste. Nessas empresas a principal técnica de teste utilizada é a execução de Testes Exploratórios.

Na maioria das vezes os testes exploratórios são realizados de maneira informal, se baseiam na intuição e dependem da habilidade e experiência do testador.

Este trabalho propõe boas práticas e uma metodologia para a execução de testes exploratórios com o intuito de melhorar os resultados obtidos com a realização desta técnica.

Palavras-chave:

Testes de Software, Testes Exploratórios, Boas Práticas, Metodologia.

Agradecimentos

Este trabalho, que foi desenvolvido durante aproximadamente quatro meses, é fruto dos conhecimentos que adquiri durante quatro anos e meio no curso de Ciências da Computação e da colaboração de muitas pessoas que estiveram ao lado durante esse período. Nada que é construído é realizado por uma única pessoa. Direta ou indiretamente várias pessoas contribuem para que algo seja realizado. Neste espaço, agradeço a todas as pessoas que de alguma forma participaram da realização deste trabalho.

Deus sem sombra de dúvidas foi quem mais esteve presente durante todo esse tempo. Nos momentos bons, nos momentos difíceis, em qualquer momento ele sempre esteve comigo. Agradeço-o por cada dia concedido, por cada oportunidade recebida, pelas experiências boas e ruins, pelas pessoas que ele tem colocado em minha vida, pela sua presença, pela sua paciência, por cada conselho e principalmente pelo seu amor. Amor que sei que nunca conseguirei entender.

Agradeço a minha mãe (minha "veinha") e ao meu pai (meu "gato") que sempre se preocuparam comigo, com o meu bem-estar. Que mesmo não acertando sempre tiveram a melhor intenção. Sempre se preocuparam em me ensinar o que é correto, princípios e valores que estão sendo deixados de lado pelo mundo, mas que são básicos para se viver bem em sociedade. Eles não mediram esforços para me educar e me dar o melhor. Deram-me bons exemplos, ensinamentos e conselhos. Não posso deixar de agradecer ao meu irmão, Ronald, que sempre se preocupa comigo e me ajuda bastante. Aos meus pais e ao meu irmão agradeço por me fornecerem uma base familiar sólida.

Agora vem a parte mais difícil, agradecer aos amigos! É um problema agradecer a todos os amigos quando são tantos. Dividirei em grupos para ficar mais fácil.

Agradeço aos amigos que tenho desde infância. "Carmita", que odeia esse apelido. Acredito que só eu ainda a chama assim. Desde que me entendo por gente a conheço, nem lembro como conheci Carmen. "Frávia", também chamada

por “vinha” ou Flávia, que quando conheci nem falava direito. Cheguei a pegá-la nos braços. Também convivo com “biscoitinho” e Jeyce desde criança. São muitos anos de amizade, respeito, companheirismo e aprendizagem. Muitos sorrisos, gargalhadas, conselhos, exortações, lágrimas.

Aline e “Carol” que são minhas amigas da escola. Bons tempos, os tempos de colégio.

Tem os meus amigos do trabalho: Aninha, Antônio, Eronys, Maíra, Osmany, Valéria e Thaís. É a equipe de teste mais divertida, prestativa e competente de Pernambuco.

Durante minha graduação fiz muitos amigos, começarei agradecendo aos que são da minha sala e estão concluindo o curso comigo. “Cara de Concha”, conhecida por Renata. Ela é a menina mais paciente que conheço. É sempre prestativa, alegre, responsável, amiga. Tem “Paty Girl”, Patrícia é uma pessoa muito inteligente e decidida. “Peo”, Pedro é o mais “tosco” de todos. Ele é muito engraçado, divertido e abestalhado. Pra fechar tem “Thiaguinho”, que admiro bastante por sua força de vontade e determinação. Nesses últimos quatro anos, foram muitos momentos juntos: manhãs, tardes, noites e às vezes madrugadas. Mas, sempre foi divertido e agradável estar com eles. Dividimos nossas alegrias e tristezas, um sempre “segurou a barra” do outro. Crescemos e amadurecemos unidos.

Existem os amigos que são de outras turmas. Passei a conviver mais com “Nana”, Nancy no meio do curso. Ela sempre tem uma palavra amiga, é uma ótima conselheira. O “Pinto” ou Marden adora me perturbar. “Kadu”, Carlos Eduardo nunca tem tempo pra nada.

No CIn aprendi muita coisa, ensinamentos que não se restringem apenas à área técnica, profissional. O CIn contribuiu para a minha formação cidadã. Agradeço aos meus professores que me passaram muitos conhecimentos e sempre estiveram dispostos a ajudar. Sou grata aos funcionários: Ana, Carlos, Diogo, Hilda, Ivanilda, Jorge, Roberto, Rosa e tantos outros. “Aninha” é como uma mãe pra mim, Pedro, Renata e Thiago. Agradeço ao pessoal do CITi e em

especial ao grupo PET Informática, aprendi bastante com eles e com meu tutor Fernando Fonseca.

Não posso deixar de agradecer ao meu orientador Alexandre Vasconcelos pela disposição, ajuda e orientações.

Por fim agradeço a você, que mesmo lendo três páginas de agradecimentos e percebendo que escrevo muito, ainda vai conseguir ler este documento até o final. Espero que lhe seja útil (esse parágrafo é um plágio do TG do meu amigo "Pinto").

Sumário

Índice de Figuras

Índice de Tabelas

1. Introdução

A importância do software tem crescido cada vez mais nas organizações e na sociedade. A cada dia ele desempenha atividades mais importantes, que trazem benefícios e agregam valor ao meio no qual atua. Geralmente, essas atividades podem ser realizadas pelo homem. Porém, algumas delas só podem ser executadas por computadores. A tendência é que o mundo se torne cada vez mais dependente dos softwares.

Determinadas atividades desempenhadas por computadores têm impacto direto na saúde financeira de uma instituição. Outras delas estão extremamente relacionadas à vida.

Torna-se cada vez mais necessário produtos com alto nível de qualidade e que satisfaçam o cliente. Por outro lado, os softwares que estão disponíveis no mercado, em sua grande maioria, apresentam uma grande quantidade de defeitos. Esses problemas afetam a funcionalidade, o desempenho, a segurança, a confiabilidade e a usabilidade do sistema, tendo impacto direto no ambiente no qual ele atua e podendo trazer conseqüências graves.

Informações de mercado dizem que mais de 90% dos sistemas são liberados com graves defeitos [1].

A insuficiência de testes é um dos principais motivos de falha nos Projetos de Desenvolvimento de Software [2].

Um processo de teste bem definido e eficiente unido a uma equipe de teste independente são fundamentais para garantir a qualidade do produto e encontrar os erros antes que a aplicação entre em produção. O custo para corrigir os erros em um sistema já implantado é bem maior que o custo que se teria para corrigir esses mesmos erros em fase de desenvolvimento.

Infelizmente, a área de testes ainda é muitas vezes negligenciada pelas organizações, que em geral, não possuem um processo de testes bem definido, eficiente, documentado e com objetivos claros, bem entendidos e mensuráveis. Algumas por não terem condições de implantar um processo como esse outras por não terem percebido a necessidade dele. Nessas instituições as atividades de

teste são informais, sem uma metodologia e responsabilidades definidas. Na maior parte das vezes essas atividades são realizadas de maneira *ad-hoc*. A principal técnica de teste utilizada é a execução de Testes Exploratórios.

Os testes exploratórios também podem ser utilizados para complementar os testes sistemáticos. Eles ajudam a encontrar novos defeitos ou cenários de teste, a entender o funcionamento da aplicação e são indicados para funcionalidades críticas e complexas.

Na maioria das vezes, para a realização das atividades de testes exploratórios não há um planejamento bem detalhado e nenhuma estratégia de teste definida. O teste exploratório se baseia na intuição e depende da habilidade e experiência do testador. A cobertura dos testes é pequena e normalmente só os defeitos podem ser reproduzidos.

É necessário tomar ações que minimizem os problemas que esta técnica apresenta, que facilitem a execução das atividades e melhorem os resultados obtidos. Uma das ações que pode ser tomada é seguir uma metodologia adequada para a realização de Testes Exploratórios.

1.1 Objetivos

O objetivo do Trabalho de Graduação apresentado neste documento é propor uma metodologia para testes exploratórios que direcione os testes, definindo objetivos e sugerindo cenários que servem de base para outros testes. Uma metodologia que planeje e organize as atividades a serem realizadas, que garanta que o sistema está de acordo com determinado padrão, que possibilite uma melhor distribuição dos recursos utilizados e um aumento na detecção dos erros.

Outro objetivo deste trabalho é apresentar boas práticas que facilitam a execução desse tipo de teste e melhoram os resultados obtidos.

1.2 Organização do documento

O restante desse documento está organizado em sete capítulos da seguinte maneira:

Capítulo 2 - Apresenta uma visão geral sobre teste de software, mostrando a evolução das atividades de testes, os conceitos, estratégias, tipos e níveis de testes e algumas formas de testar.

Capítulo 3 – Aborda o tema testes exploratórios, suas características e são expostos alguns cenários onde esses testes são aplicados.

Capítulo 4 – Dedica-se a mostrar algumas boas práticas para a realização de testes exploratórios.

Capítulo 5 - Com base no que foi apresentado nos capítulos anteriores, este capítulo propõe uma metodologia para testes exploratórios.

Capítulo 6 – Relata a utilização de uma metodologia parecida com a proposta neste trabalho em uma empresa. Esse capítulo descreve o ambiente, a aplicação da metodologia, uma análise da metodologia por quem a utilizou na prática e os benefícios para a organização.

Capítulo 7 - Apresenta as considerações finais, além de propor trabalhos futuros.

2. Visão geral sobre Testes de Software

2.1 Evolução da Atividade de Teste

A atividade de testar o software até alguns anos estava em segundo plano. Era uma atividade que não recebia muita atenção e nem investimentos. Não se investia nem em tempo e nem em recurso para a realização dessa tarefa. Há poucos anos não se encontrava muito sobre o assunto na literatura.

Em 1950, Alan Turing escreveu o primeiro artigo científico que abordava questões sobre teste de software. Ele definia um teste operacional para demonstrar o comportamento da inteligência de um programa semelhante ao de um ser humano.

Antes de 1957, teste era a simples tarefa de navegar pelo código e corrigir erros já conhecidos. Foi em 1957 que o conceito Teste de Software se tornou um processo de detecção de erros de software. Mas, essa atividade só ocorria no final do processo de desenvolvimento.

Quando a Engenharia de Software, no início da década de 1970, passou a ser utilizada como modelo para organizações e universidades o processo de desenvolvimento de software passou a ter abordagens mais profundas. Em 1972 ocorreu a primeira conferência sobre testes na Universidade da Carolina do Norte.

Mas, um dos primeiros trabalhos mais completos e profundos sobre um processo de teste só foi produzido em 1979 por Glenford Myers. Foi nesse trabalho que a atividade de teste foi definida como um "processo de trabalho com a intenção de encontrar erros". Até esse momento, o objetivo do teste era somente provar o bom funcionamento de uma aplicação. Todos os esforços da atividade estavam voltados para a comprovação desse fato e conseqüentemente poucos defeitos eram encontrados. Myers propôs que se o objetivo do teste for encontrar erros, uma quantidade maior de problemas será encontrada. Uma vez que vários cenários serão buscados para testar o comportamento do aplicativo em várias circunstâncias.

Essa nova visão revolucionou a forma de abordar o problema, no entanto, os testes continuavam sendo executados no final do processo de desenvolvimento.

Os primeiros conceitos de qualidade de software surgiram no início dos anos 1980. Nesses novos conceitos, desenvolvedores e testadores trabalham juntos desde o início do processo de desenvolvimento.

A partir deste período, teste passou a ser visto como um processo paralelo ao processo de desenvolvimento. Passou a possuir atividades específicas de planejamento, análise de requisitos dos testes, design, implementação, execução e manutenção de testes [3].

As primeiras ferramentas para realizar as atividades teste só começaram a ser fabricadas em 1990. Elas possibilitam a execução de teste que não podiam ser feitos manualmente. Como testes de carga, performance, entre outros.

A visão de testes tem evoluído para uma atitude mais pró-ativa, não se limitando apenas à detecção de falhas, mas atuando fortemente na prevenção [3].

2.2 *Teste de Software*

Atualmente, o mercado está cada vez mais exigente competitivo e as empresas buscam estratégias para sobreviver. Melhorar a qualidade do produto final e reduzir os custos do desenvolvimento deste produto são importantes objetivos das organizações. O processo de teste de software visa aumentar a qualidade do produto, reduzir os custos do desenvolvimento e de manutenção, reduzir os defeitos dos produtos, diminuir o esforço e o custo de retrabalho, aumentar a confiabilidade do produto e, consequentemente aumentar a satisfação do cliente. Os testes são as últimas oportunidades de se encontrar defeitos antes que o produto entre em produção.

Vários autores apresentam uma definição para testes de software, algumas delas são:

Testar um software significa verificar através de uma execução controlada se o seu comportamento corre de acordo com o especificado. O objetivo principal desta tarefa é encontrar o máximo número de erros dispondo do mínimo de

esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos [4].

Teste é um processo sistemático e planejado que tem por finalidade única a identificação de erros [5].

Operação de um sistema ou aplicação, em um ambiente controlado, de situações normais e anormais, e a avaliação dos resultados para verificar se o sistema se comporta conforme previsto [6].

Segundo a *IEEE Standard Glossary of Software Engineering Terminlogy*, teste de software é um processo de exercitar um software ou componente sobre condições específicas, observando ou gravando os resultados, e realizando uma avaliação sobre algum aspecto.

Os testes podem ser usados para mostrar a existência de erros, mas nunca a ausência deles. Por mais que você teste sempre existirão erros, zero-defeito é algo inatingível [7]. É impossível ter um produto livre de erros devido ao número enorme de situações existentes, cenários possíveis. Depois de um determinado tempo testando a probabilidade de encontrar erros diminui bastante. O momento certo de parar de testar é o momento em que o custo para testar e corrigir os erros é mais caro que o custo da falha ocorrer em produção [11]. A Figura 1 mostra o momento de interromper os testes.

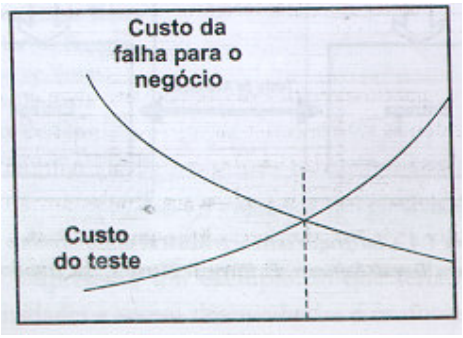

Figura 1: Gráfico mostrando o momento de interromper os testes

A criticidade do sistema é quem define a quantidade de testes. Quanto mais crítico for o sistema mais testes terão que ser executados para garantir a qualidade dele.

Todo tipo de erro custa dinheiro. Além da alocação de recursos e tempo para produzir algo com defeito, existem os custos que o erro provoca como:

identificação e correção do erro, implantação da correção. Segundo Myers, quanto mais tarde um erro é descoberto mais caro ele é. Ele aplica a "regra de 10" aos custos de correção do erro. Ou seja, quando um erro não é encontrado em uma fase do processo de desenvolvimento, os custos de correção são multiplicados por 10 para cada fase que ele se propaga. Isso significa dizer que erros encontrados em produção são extremamente caros.

2.3 *Verificação e Validação*

As atividades de teste começam no início do processo de desenvolvimento com os testes de verificação. Teste de software inclui atividades de verificação e validação.

A verificação diz respeito ao que foi construído, se a aplicação foi construída corretamente. Já a validação diz respeito ao entendimento do que era pra ser construído, se foi construído o sistema correto. A figura 2 exemplifica bem o que acontece durante a construção de um sistema.

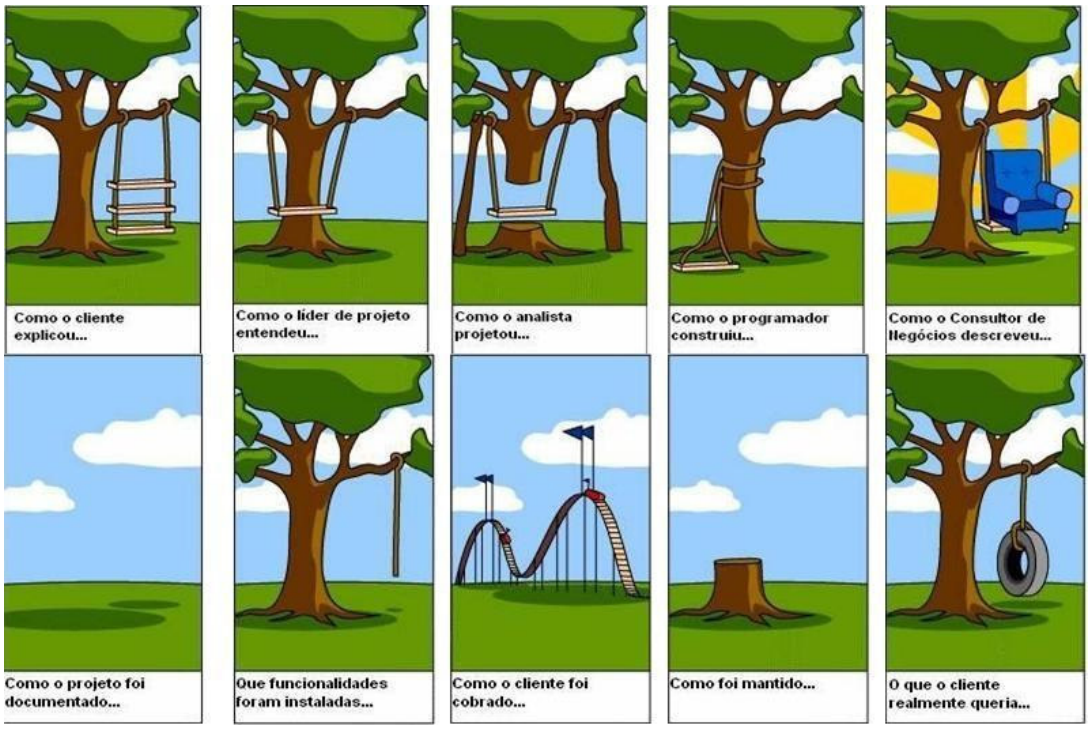

Figura 2: Validação x Verificação

Testes de Verificação

Esse tipo de teste também é conhecido como teste estático, pois geralmente são realizados sobre os documentos que são produzidos durante todo o ciclo de desenvolvimento. Essas atividades são iniciadas antes da codificação do sistema, elas começam na fase de requerimento e continuam através da codificação do produto. O teste consiste na realização de inspeções, revisões e/ou auditorias sobre os produtos gerados em cada etapa do processo, evitando que dúvidas e assuntos mal resolvidos passem para a próxima fase do processo de desenvolvimento. Algumas atividades de verificação são:

- Revisões de requisitos;
- Revisões de modelos;
- Revisões e inspeções técnicas em geral.

A verificação tem por objetivo provar que o produto vai ao encontro dos requerimentos especificados. Ela garante a qualidade do software na produção e na manutenção.

Testes de Validação

Os testes de validação também são conhecidos como testes dinâmicos ou testes de software, uma vez que são aplicados diretamente no software. Eles podem ser aplicados em componentes isolados, módulos existentes ou em todo o sistema. Esse tipo de teste avalia se o sistema atende aos requisitos e especificações analisadas nas fases iniciais do projeto, se ele vai ao encontro dos requerimentos do usuário. Algumas atividades de validação são:

- Teste unitário
- Teste de integração
- Teste de sistema
- Teste de aceitação
- Homologação
- Teste de Regressão.

Testes de verificação e validação são complementares, eles não são atividades redundantes. Eles possuem objetivos e naturezas distintas, contribuem para a detecção de erros e aumentam a qualidade final do produto.

2.4 Estratégias Fundamentais dos Testes

Para executar os testes de validação existem duas estratégias: estratégia caixa branca e a estratégia caixa preta. A estratégia determina como os testes serão conduzidos e é aplicada para diminuir esforços e aumentar a probabilidade de detecção de erros presentes no produto.

Essas estratégias não são excludentes, elas são complementares. Se as duas estratégias forem combinadas de forma apropriada o resultado final será um software de melhor qualidade.

Estratégia Caixa Branca

Os testes caixa branca são baseados na arquitetura interna do software, por isso, são conhecidos como caixa branca. Eles identificam defeitos nas estruturas internas do programas através da simulação de situações que exercitem adequadamente todas as estruturas utilizadas na codificação. Logo, é necessário que o testador tenha conhecimento suficiente sobre a arquitetura interna e conheça a tecnologia utilizada no software.

O objetivo desses testes é garantir que todas as linhas de código e condições foram executadas pelo menos uma vez, e estão corretas. Alguns métodos são aplicados nos testes de caixa branca. Exemplo: Método de Cobertura de Linhas de Código, Método de Cobertura de Caminhos, Métodos de Cobertura de Desvios Condicionais, Método de cobertura de Laços.

Esses testes são difíceis de implementar e são altamente eficientes na detecção de problemas. Uma característica desse tipo de teste é que ele pode ser modelado, estruturado e executado pelos desenvolvedores.

Estratégia Caixa Preta

Essa estratégia utiliza técnicas para garantir que o software atende os requisitos do sistema. Esses testes não verificam os processamentos internos, se a

solução adotada foi a melhor, mas verificam se o sistema produz o resultado esperado. Ela não se preocupa com o código, testa as entradas e as saídas da aplicação.

Para adoção dessa estratégia não é necessário conhecer a tecnologia utilizada, a arquitetura do sistema. Isso facilita a modelagem dos testes. O que é necessário é o conhecimento dos requisitos, o comportamento esperado do sistema para que se possa avaliar se o resultado produzido é o correto.

Eles são conhecidos como mais simples de implementar que os de caixa branca.

2.5 *Tipos de Testes de Software*

Segundo Leonardo Molinari, tipo de teste diz respeito ao que tem que ser testado [8]. Os tipos de teste que ele apresenta são:

Teste Funcional

Para a execução desse tipo de teste é necessário conhecimento das regras de negócio do sistema para que os cenários de negócio sejam simulados. O teste verifica se as funcionalidades se comportam de acordo com a documentação de especificação funcional, ou seja, se não existe diferença entre os requisitos funcionais e o comportamento da aplicação. Valores de entrada e saída são utilizados para a avaliação do comportamento do software. Esses testes são caracterizados pelo foco em negócios.

Teste de Interface

Esses testes focam na interação do usuário com o sistema. Eles verificam o acesso e a navegação apropriados através das funções da aplicação e o funcionamento correto dos objetos da tela.

Teste de Performance

Os testes de performance verificam se o tempo de resposta de uma determinada situação é condizente com tempo especificado nos requisitos definidos. Eles analisam o sistema sob condições reais de acesso simultâneo ou

de carga de dados. As situações de pico máximo previstas são comparadas como os valores-limite especificados nos requisitos. Em geral teste de performance e de carga se confundem, mas podemos dizer que o de performance se refere à medição de tempo.

Teste de Carga

Tipo de teste utilizado para validar que o sistema suporta a quantidade de usuários requeridos. Ele avalia o comportamento do sistema para um volume de usuários simultâneos que simule uma quantidade real.

Teste de Estresse

O teste de estresse é destinado a avaliar como o software se comporta em condições anormais, testa o aplicativo em situações inesperadas. Visa ir além dos limites do sistema, seja de usuários simultâneos, seja de dados, seja de processos na rede. O teste pode envolver cargas extremas de trabalho, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados.

Teste de Volume

Esses testes dão enfoque a parâmetros de sobrecarga. Ao ser testada a aplicação é submetida a grandes quantidades de dados. Nesse teste o volume de dados é incrementado sucessivamente até que o limite máximo que a infra-estrutura está preparada para processar seja atingido. O objetivo é conhecer os limites do software e comparar com os requisitos especificados.

Teste de Configuração

Basicamente, esses testes verificam se a aplicação funciona como esperado em diferentes ambientes de software e hardware, ou seja, o aplicativo é testado em várias configurações de software e hardware. Esses diferentes ambientes são os ambientes previstos na fase de elicitação de requisitos.

Teste de Instalação

Os testes de instalação têm por objetivo garantir que a instalação do sistema ocorrerá corretamente em diferentes ambientes de software e hardware e em diversas condições (falta de energia, falta de espaço em disco). Recomenda-se que este teste seja executado pelo usuário.

Teste de Documentação

Como o próprio nome diz, esses testes focam na documentação do sistema. Os testes analisam a completude e a clareza dos documentos.

Teste de Integridade

Tipo de teste que verifica a robustez, a resistência à falhas do objeto do teste e a integridade dos dados armazenados.

Teste de Segurança

Esses testes têm a finalidade de encontrar as falhas na segurança do sistema que podem provocar perdas de dados, interrupções de processamento, comprometimento do sigilo e fidelidade das informações.

Os ataques à segurança do sistema podem ter origens internas ou externas e podem ser realizados por profissionais descontentes, quadrilhas especializadas, *hackers*.

2.6 Estágio ou Nível de Testes

O estágio ou nível de teste diz respeito a quando testar, em que fase do processo de desenvolvimento determinado teste deve ser feito.

Teste de Unidade

O objetivo desses testes é testar um pedaço do código. Os testes são aplicados nos menores componentes de código e testam unidades individuais como: funções, classes, componentes.

Os testes de unidade podem testar a estrutura interna, funcionalidade, segurança dos componentes. Para a realização desses testes é requerido um

conhecimento da estrutura interna do software. Geralmente esses testes são realizados por desenvolvedores.

Teste de Integração

Nesses testes são avaliadas as interações entre partes do software. São realizados no final de cada iteração e comprovam que duas ou mais funções, componentes juntos funcionam corretamente. Esse teste requer um conhecimento da arquitetura interna do software. Eles podem testar interfaces e dependências entre componentes.

Teste de Sistema

Os testes de sistema são executados no sistema como um todo e em um ambiente teste, mas esse ambiente tem que ser muito parecido com o de produção. Não é necessário conhecimento da arquitetura interna do software. Os testes devem ser executados por uma equipe de testes independente. Nesses testes tanto os requisitos funcionais quanto os não funcionais podem ser testados.

Teste de Aceitação

Testes feitos pelos usuários finais. Os testes são realizados no sistema como um todo. O usuário valida se o sistema está pronto e pode ser usado por usuários finais, se as funções do sistema estão funcionando corretamente. A usabilidade e a segurança do software também podem ser testadas. Geralmente esses testes são realizados em ambiente de homologação.

Para a realização desses testes não é necessário conhecimento da estrutura interna do sistema.

2.7 Formas de Testar

As atividades de teste podem ser realizadas basicamente de duas formas:

Testes Manuais

Na maior parte das empresas os testes são manuais. Os testes manuais podem ser divididos em: testes sistemáticos e testes exploratórios.

- Testes Sistemáticos

Testes predefinidos executados seguindo os passos do Projeto de Teste. Projeto de Teste é um conjunto predefinido de passos e resultados esperados que são baseados nas especificações do projeto.

Esses testes geralmente são empregados em instituições que seguem um processo de teste e possuem uma equipe de testes independente. Esses testes são recomendáveis para projetos aderentes a um processo de desenvolvimento, que apresentação documentação de qualidade, além de tempo e recurso disponíveis. Esses testes verificam formalmente se o sistema está de acordo com a sua especificação, permitem identificar as dependências entre os testes através da matriz de rastreabilidade. Nesse tipo de teste as atividades acontecem em paralelo com as atividades de desenvolvimento. Outras características desses testes são: elaborados por pessoas que possuem o perfil de testador, alta cobertura dos testes e são facilmente reproduzidos. Qualquer pessoa que for testar o produto usando testes sistemáticos vai executar os mesmos testes e da mesma forma, uma vez que o passo-a-passo para execução dos testes está documentado.

- Testes Exploratórios

O teste exploratório é, na sua definição mais básica, a criação e a execução ao mesmo tempo de um teste [9].

São testes em que o testador controla ativamente o design dos testes durante a execução e utiliza informações adquiridas para projetar novos e melhores testes [10]. No próximo capítulo, essa forma de testar será mais detalhada.

Testes Automáticos

A automação dos testes é a automação dos casos de testes, a geração automática da massa de dados de testes e execução automatizada dos testes.

Essa automação inclui a automação do processo de planejamento dos testes, permite aumentar a profundidade e abrangência dos casos de testes envolvidos, viabiliza a execução de alguns tipos de testes, como é o caso dos

testes de carga e estresse. Porém, os projetos de automação de teste devem ser implementados com muito cuidado. A seleção da ferramenta de automação de teste bem como o que deverá ser automatizado ou não devem ser analisados de forma clara porque dependendo das técnicas de teste utilizadas, o retorno de investimento pode ser negativo ou positivo.

Algumas expectativas em relação à automação são criadas, mas nem todas são verdadeiras, exemplo:

- Todos os testes podem ser automatizados;
- Uma única ferramenta de automação atende todos os tipos de teste;
- Automação é uma alternativa para testar um sistema pouco documentado;
- Automação é uma alternativa para testar um sistema mal planejado;
- Ao automatizar a redução de esforço e tempo para realização das atividades de teste é imediata;
- Ferramentas de automação de testes são fáceis de usar.

A automação só deve ser realizada quando existe um processo maduro na organização e na equipe de testes, conscientização por parte da organização e os testes levam muito tempo para serem executados manualmente. Os testes automáticos são indicados para testes de produto, testes com múltiplos usuários, testes com múltiplas configurações de hardware e software.

2.8 Considerações Finais

Este capítulo apresentou uma visão geral sobre testes de software e introduziu uma base teórica para entender o próximo capítulo que aborda o tema testes exploratórios.

3. Testes Exploratórios

Um processo de teste bem definido e eficiente unido a uma equipe de teste independente são fundamentais para garantir a qualidade do produto e encontrar os erros antes que o software entre em produção.

O objetivo de um processo de testes com metodologia própria é minimizar os riscos causados por defeitos provenientes do processo de desenvolvimento [11].

Infelizmente, muitas organizações não possuem um processo de teste bem definido e eficiente, sendo as atividades de teste informais, sem uma metodologia e responsabilidades definidas. Na maior parte das vezes essas atividades são realizadas de maneira *ad-hoc*. Ainda hoje teste é uma atividade que todo mundo sabe que precisa fazer, mas que ninguém quer fazer. Ela é considerada uma atividade chata, que consome tempo e que na maioria das vezes só é valorizada quando o produto entra em produção e os mais variados erros acontecem.

Para os casos onde o processo de teste não é bem definido, a principal técnica de teste utilizada é a execução de Testes Exploratórios.

Como definido no capítulo anterior, os casos de teste são projetados, executados dinamicamente no momento da execução. Na maioria das vezes eles são criados de maneira intuitiva. Poucos casos de testes são executados. Normalmente, o próximo caso de teste se baseia no anterior.

As atividades em testes exploratórios ocorrem quase em paralelo. Não existe nenhum planejamento explícito dessas atividades. A pessoa que executa o teste também é o projetista dos testes.

Os testes exploratórios ajudam muito quando o projeto não contém documentação especificando o sistema ou esta é de baixa qualidade ou desatualizada. Essa técnica traz resultados rápidos, sendo assim, indicada quando o prazo é curto.

Testes exploratórios são indicados nas seguintes situações [10]:

- Fornecer rápido feedback de um novo produto ou nova funcionalidade;

- Conhecer o sistema num curto espaço de tempo;
- Já utilizou testes baseados em caso de uso;
- Investigar o trabalho de outro testador;
- Investigar e isolar um defeito particular;
- Investigar o estado de um risco particular, avaliando a necessidade de uma execução de testes baseada em casos de uso desta funcionalidade.

Para executar um teste exploratório normalmente o testador realiza algumas ações: conhece o produto, aprende os pontos fracos do produto, aprende o caminho em que o produto falha, aprende a testar o produto, reporta os problemas, faz novos testes com o que foi aprendido até o momento.

Os testes exploratórios não são nem caixa preta e nem caixa branca, pois não se baseiam nem nos requisitos e nem no código fonte do projeto.

3.1 Testes Exploratórios Realizados por uma Equipe de Testes

Este cenário pode ser aplicado a uma empresa que possui um processo de teste independente do processo de desenvolvimento. E o processo de teste é seguido por uma equipe de teste independente. Mas, nessa organização nem todos os projetos estão dentro do processo de desenvolvimento.

Neste cenário o teste exploratório é indicado para o projeto que não contém documentação especificando o sistema ou esta é de baixa qualidade ou desatualizada, onde os recursos são limitados e o prazo é curto. Ou para projetos que estão em fase de manutenção.

A aplicação de testes exploratórios traz resultados rápidos e consome pouca quantidade de recursos.

Para testar a aplicação o testador cria casos de testes para descobrir problemas esperados e suas causas. Ele cria os casos com base no seu conhecimento, habilidade e experiência. Utilizando esses recursos ele sabe onde provavelmente erros irão ocorrer. Ele também pode utilizar a experiência de ter testado sistemas parecidos para encontrar erros com mais facilidade.

Como o testador, geralmente, não conhece bem o sistema, as regras de negócio. Essa falta de conhecimento requer um maior tempo para entendimento do negócio e consequentemente, o tempo para realizar os testes torna-se maior.

Ao término dos testes, o sistema é comparado com as expectativas e o entendimento do testador. Ou seja, os casos de teste dependem muito do testador. A cobertura dos testes é baixa e existe a tendência a encontrar um número reduzido de falhas. Geralmente só as falhas podem ser reproduzidas e não existe nenhuma evidência de quais partes do sistema forma testadas.

3.2 Testes Exploratórios Realizados por Especialistas

Este cenário é adequado para empresas que não possuem um processo de testes definido. Normalmente, nessas organizações os testes são executados por especialistas no sistema, analistas e desenvolvedores.

Este cenário é indicado para projetos nos quais as funcionalidades do sistema são pouco documentadas, o prazo é curto e os recursos são limitados.

A vantagem é que os testes são executados por pessoas que conhecem o que sistema deve fazer, como ele deve se comportar. O tempo e a equipe para essas atividades de testes são reduzidos. O problema é que neste cenário os testes são executados por pessoas que não são da área de teste, que não possuem o perfil de testador. Elas não foram treinadas e nem capacitadas para exercer as atividades de teste. A grande parte dos analistas e desenvolvedores não gosta de testar, testam porque é necessário. Na maioria das vezes testam para provar que o sistema está funcionando corretamente, não para encontrar erros.

Geralmente os testes são viciados, são associados apenas ao fluxo principal do sistema e normalmente só valores válidos são utilizados. A quantidade de falhas encontradas é reduzida. A cobertura dos testes é bem menor do que a alcançada com testes baseados em caso de uso.

Ao término dos testes, como no cenário anterior, o sistema é comparado com as expectativas e o entendimento do testador, na maioria das vezes apenas os defeitos são reproduzidos e não existe nenhum registro de que parte do sistema foi testada.

3.3 *Testes Exploratórios aliados a Testes Sistemáticos*

Nada impede que os testes exploratórios sejam executados em paralelo com os testes baseados em roteiros. Isso é até uma prática muito indicada.

Os testes exploratórios ajudam a encontrar novos defeitos ou cenários de testes, a entender o funcionamento da aplicação. Também podem complementar os casos de testes baseados em roteiro antes dos testes de homologação. Eles detectam erros que não poderiam ser encontrados baseando-se apenas na especificação do projeto. E é indicado para funcionalidades críticas e complexas.

3.4 *Considerações Finais*

Este capítulo explanou o tema testes exploratórios, apresentou conceitos e características, vantagens e desvantagens da utilização dessa técnica. Além de mostrar cenários onde ela pode ser aplicada. Para minimizar as desvantagens encontradas na utilização desta técnica o próximo capítulo irá listar algumas boas práticas para a realização dessa técnica e o capítulo cinco irá propor uma metodologia para execução das atividades de teste exploratório.

4. Boas práticas para Testes Exploratórios

Neste capítulo serão apresentadas algumas boas práticas que podem facilitar a execução desse tipo de teste.

4.1 Boas Práticas

Algumas boas práticas que podem ser utilizadas são:

Estudar Problemas de Sistemas Anteriores ou Similares

Pode-se levantar os problemas encontrados com os sistemas anteriores ou similares através de:

1. Entrevistas com usuários para descobrir as falhas do sistema anterior
2. Leitura sobre sistemas similares

Leitura de Documentos de Referência

Ler documentos que expressam o que é o sistema, assistir a vídeos ou apresentações que falam o que é o sistema. As informações obtidas servirão como entrada para a criação dos testes.

Essa prática se aplica quando os testes são executados por uma equipe de testes independente. Os especialistas, os desenvolvedores e os analistas, já conhecem o sistema.

Treinamentos realizados por Especialistas

O líder de projeto pode dar um treinamento sobre o que é o sistema, fazer uma pequena apresentação mostrando o que é o sistema e suas principais funcionalidades, quais as funcionalidades mais críticas, o que deve ser testado.

Como a prática anterior, esta é indicada quando os testes são executados pela equipe de testes independente.

Treinamentos pela Equipe de Teste

Quando os testes são executados por desenvolvedores, analistas, a equipe de teste pode dar um treinamento sobre testes, testes exploratórios. Minimizando assim, os testes viciados e direcionado os testadores.

Utilização de Heurísticas

Heurísticas iniciais podem auxiliar a execução dos testes, gastando assim menos esforço. Testes podem ser utilizados para aprender o básico do produto.

Aconselha-se testar os fluxos básicos primeiro, as funcionalidades isoladas antes de combinar as funcionalidades.

Lista de Modos de Falha

Cria-se uma lista que contém modos de falha para a organização. Existem várias listas disponíveis que fornecem modos de falha comuns a vários projetos. Exemplo de modo de falha: arquivo necessário não encontrado no lugar especificado.

Desenvolvimento de Cenários

Vários cenários para testes podem ser criados. Para criar esses cenários pode-se:

1. Escrever histórias reais para os objetos do sistema. Exemplo: Após o cadastro de um usuário na aplicação, o sistema envia um e-mail informando o login e a senha do usuário.
2. Listar os usuários do sistema e o acesso de cada perfil, testar se eles têm acesso às funcionalidades esperadas.

Ataques

Existem alguns tipos de defeitos que são comuns a tantos sistemas que existem ataques padrões para eles. Existe até um livro que trata sobre esse assunto, “How to break software” Whittaker. Exemplos:

1. Usar todos os tipos de dados e caracteres permitidos;
2. Utilizar tipos de dados e de caracteres inválidos;

3. Estourar os limites de entrada;
4. Explorar entradas que fazem com que todas as mensagens de erro apareçam.

Testes em Par

Teste em par foi baseado em uma das práticas de XP, a programação em par. Os testes em par consistem na execução do teste por dois testadores que utilizam o mesmo computador. Como em XP, os pares não são fixos e podem mudar ao longo do dia, essas mudanças são importantes para a disseminação do conhecimento. Os testes em par contribuem para um espaço de aprendizado contínuo dentro da equipe. Os mais experientes ou quem tem mais conhecimento sobre um assunto termina passando informações valiosas para seus pares. Todos aprendem o que há de melhor com seus colegas. Os testadores aprendem as coisas boas com seus colegas, podem ensinar e também corrigem aquilo que seu par faz de errado. Logo, essa prática procura potencializar o que existe de melhor em cada um e eliminar as falhas.

A idéia é que enquanto um testador executa o teste o outro presta atenção, faz perguntas, sugere idéias, toma nota, etc. Essa prática pressupõe uma comunicação contínua entre os testadores que formam o par. Através da conversa eles analisam cenários, discutem as melhores alternativas.

Esse tipo de prática ajuda o testador a se manter focado no teste. Um *insight* não precisa ser interrompido para se tomar nota, tirar dúvidas, consultar manual ou documentação, replicar o erro em outra máquina. Essas atividades podem ser realizadas pelo outro testador. Ela também melhora a reportagem dos erros, pois facilita a reprodução e tudo que é reportado é revisado por uma outra pessoa. Além de ser um bom treinamento para novatos. Existe pelo menos um conhecimento técnico e um conhecimento sobre as informações do projeto. Testes em par permitem que alguém que acabou de entrar na equipe conheça rapidamente o sistema, pois o conhecimento é transmitido pelo parceiro à medida que a atividade é executada.

Precisa-se ter cautela ao escolher as pessoas para executar essa atividade. Algumas pessoas são introvertidas, precisam de tempo trabalhando sozinhas para desenvolver idéias. Pode existir conflito entre o par e o responsável pela escolha dos pares deve sempre estar atento para detectar problemas e trocar os pares sempre que necessário. E a estrutura física do ambiente de trabalho deve oferecer espaço suficiente para que duas pessoas trabalhem juntas em um mesmo computador.

Utilização de um Guia de Cenários

A utilização de um guia facilita e direciona a execução dos testes realizada por desenvolvedores, analistas. Esse guia deve conter os principais cenários de testes. Além de aumentar consideravelmente a cobertura dos testes, esse guia serve de base para outros testes e garante que qualquer pessoa que for testar o sistema executará no mínimo aqueles testes.

Planilha de Execução para Testes Exploratórios

Outra forma de facilitar a execução dos testes e de certa forma padronizar os testes que são executados por diversos testadores é a adoção de uma planilha de testes exploratórios. Nessa planilha existirá a maior parte dos cenários de testes que podem ser executados para uma determina funcionalidade, campo para indicar se o cenário passou ou falhou, campos para os registros abertos. Também é importante ter as seguintes informações: sistema, versão do sistema, funcionalidade, executor, data de execução, tempo total gasto, registros abertos. Além dos resultados: total de cenários, total de cenários passados, total de cenários falhos.

Antes de começar a execução dos testes o líder ou mesmo o testador prepara a planilha. Durante a execução o testador lê o cenário, excuta o teste e informa na planilha o resultado do mesmo. Assim, padronizam-se os testes mínimos que devem ser executados, tem-se o histórico dos testes que foram executados e se passaram ou não, sabe-se quem executou aqueles testes e o tempo de execução do teste. Esse tempo pode ser utilizado como entrada para as estimativas de esforço.

Documentar os erros

Pode-se criar um projeto de teste à medida que os testes são executados. Ao encontrar um erro, cria-se um caso de teste para o mesmo. O projeto é incrementado a cada erro encontrado. Assim, quando os re-testes forem executados o testador não terá problemas para executar os testes.

4.2 Considerações Finais

Este capítulo listou algumas boas práticas para a realização de testes exploratórios. Com base em algumas boas práticas apresentadas o próximo capítulo propõe uma metodologia para testes exploratórios.

5. Metodologia para Testes Exploratórios

Nesse Capítulo são sugeridos alguns passos básicos para melhorar os resultados obtidos com a prática da execução de testes exploratórios. Aqui, uma metodologia para a execução de testes exploratórios é proposta. Essa metodologia deve ser adequada à realidade de cada organização que irá utilizá-la, cada fase deve ser adaptada.

5.1 Planejamento

Esta é a primeira fase. Nela o Plano de Teste é elaborado com base no Plano de Projeto. Este documento deve conter as seguintes secções:

Escopo dos Testes

É nesta seção que o escopo dos testes é delimitado. Ela relata quais os requisitos funcionais e não-funcionais serão testados.

Cronograma

No cronograma está especificado quando e quanto tempo vai durar cada fase do processo de teste. O cronograma de testes deve estar alinhado com o cronograma do projeto.

Papéis e Responsabilidade

Os papéis e as responsabilidades dentro da execução de cada fase são definidos aqui. Também é especificado quem da equipe irá desempenhar determinado papel.

Estratégia

Aqui são definidas as estratégias que serão utilizadas. Quais técnicas serão usadas para testar determinadas funcionalidades.

Ambiente

Esta seção contempla a definição do ambiente necessário para a execução dos testes.

O Plano de Teste deve estar sempre atualizado de acordo com o Plano de Projeto. O modelo para este documento não foi criado para que cada instituição tenha liberdade para definir seu próprio modelo.

5.2 Elaboração de Cenários Básicos

Com base no Guia de Cenário de Testes, que contém uma quantidade de cenários que se aplica a várias funcionalidades, são selecionados os cenários que se aplicam às funcionalidades que serão testadas e é montada a planilha de execução.

Para montar cenários de testes algumas técnicas podem ser utilizadas:

Partição de Equivalência

O domínio dos dados de entrada é separado em classes para evitar casos de testes redundantes. A divisão das classes é baseada na possível identificação de um erro. Exemplo:

Em um sistema que contém um cadastro de produtos onde o campo descrição do produto deve conter de 3 a 50 caracteres. As classes identificadas são: quantidade de caracteres abaixo de três, quantidade de caracteres de 3 a 50 e quantidade de caracteres acima de 50. Em cada uma dessas classes existe um conjunto de valores que provavelmente possuem a mesma capacidade de encontrar erros. Logo, pode-se executar um teste para cada classe de equivalência e obter o mesmo resultado que seria encontrado ao executar vários testes para cada uma das classes. A tabela 1 ilustra a partição de equivalência.

Entrada	Valores Permitidos	Classes	Cenários
Descrição do Produto	De 3 a 50 Caracteres	< 3	Quantidade de caracteres = 1
		3 a 50	Quantidade de caracteres = 20
		> 50	Quantidade de caracteres = 60

Tabela 1: Partição de Equivalênica

Valores-limite

O método valores-limite é complementar à partição de equivalência. Em partição de equivalência qualquer valor da classe pode ser utilizado, aqui os valores-limite são os indicados. Acredita-se que a aplicação apresenta mais defeitos nos limites das classes que nas áreas centrais. Portanto, usar valores-limite nos testes aumenta a probabilidade de se encontrar problemas. A tabela 2 mostra os cenários que temos utilizando valores-limite no exemplo anterior:

Entrada	Valores Permitidos	Classes	Cenários
Descrição do Produto	De 3 a 50 Caracteres	< 3	Quantidade de caracteres = 2
		3 a 50	Quantidade de caracteres = 3 Quantidade de caracteres = 50
		> 50	Quantidade de caracteres = 51

Tabela 2: Valores-limite

Valores Brancos ou Nulos

Esta técnica consiste em não preencher as entradas que são obrigatórias para o sistema.

Valores Inválidos e Negativos

São utilizados dados inválidos nas entradas do software. O processamento desses dados inválidos geralmente provoca erros. Exemplo: Digitar caracteres em um campo numérico.

Outra situação é o caso de alguns sistemas que permitem que campos que só devem aceitar números positivos armazenem números negativos. Exemplo: O sistema permite que o valor de um produto seja -100,00.

Nem sempre os sistemas permitem digitar valores inválidos e/ou negativos, mas muitas vezes eles permitem que esses valores sejam colados. Se um sistema não permite que você digite o valor desejado, tente copiar e colar o valor desejado no campo de entrada. Isso funciona muitas vezes.

Combinação de Dados

São feitas várias combinações de dados de entrada. Cada combinação deve produzir um resultado específico. Nas combinações podem ser utilizados valores corretos, valores errados, valores inválidos, valor vazio.

Guia de Cenários de Teste

Com base nas técnicas citadas para definir cenários foi montado um Guia de Cenários de Teste para execução de Testes Exploratórios. O guia proposto nesse trabalho é um guia para execução de testes de sistema. O guia é genérico e pode ser empregado em vários projetos, para testar várias funcionalidades. Ele é divido em testes de campo, usabilidade, mensagem, negócio e segurança. A figura 3 mostra o guia desenvolvido.

Guia de Cenários de Testes

	Cenários	Exemplo	Observações
CAMPO	VALIDAÇÃO DO PREENCHIMENTO DO CAMPO	Valor permitido de 3 a 50	
	1. Valor maior que valor limite	quantidade de caracter igual a 51	
	2.. Valor igual ao valor limite	quantidade de caracter igual a 50	
	3 Valor menor que valor limite	quantidade de caracter igual a 25	
	4. Valor igual ao valor minimo	quantidade de caracter igual a 3	
	5. Valor menor que valor minimo	quantidade de caracter igual a 2	
	6. Em branco (não preenchido)		
	7. Valor maior que valor limite (CtrlC+CtrlV)		Copiar e colar uma quantidade de caracteres maior que a permitida
	8. Espaçamento entre Caracteres		
	VALIDAÇÃO DO CAMPO DO TIPO NÚMERO		
	1. Caracteres especias	#$%¨&	
	2. Números negativos	-6	
	3. Números decimais	6,78	
	4. Caracteres Alfanuméricos	5fg7	
	VALIDAÇÃO DO CAMPO DO TIPO DATA		
	1. Mês inexistente	18/00/2007	
	2. Dia inexistente	32/01/2007	
	3. Dia negativo	-1/01/2007	
	4. Mês negativo	18/-1/2007	
	5. Ano negativo	18/01/-2007	
	6. Caracteres alfanuméricos	dd/mm/2007	
	7. Formato inválido	18/1/2007	
	8. Data maior que atual	data de amanhã	Quando a maior data permitida é a atual
	9. Data menor que a atual	data de ontem	Quando a menos data permitida é a atual
	10. Data final menor que a data inicial		Quando existe data de início e dara de término
	11. Ano bissexto	29/2/2008	
	VALIDAÇÃO DO CAMPO DO TIPO HORA	Formato HH:MM	
	1. Hora inválida	25:25	
	2. Minuto inválido	12:67	
	3. Hora negativa	-6:57	
	4. Minuto negativo	13:-8	
	5. Hora vazia	:34	
	6. Minuto vazio	15:	
	7. Formato	12:34:23	
	VALIDAÇÃO DE OUTROS CAMPOS		
	1. Validação do campo email	exemplo#exemplo.com	sem @
	2. Validação campo CPF	046.567.345-78	com valores inválidos
	VALIDAÇÃO DO CAMPO (OUTROS)		
	1. Somente leitura e/ou editáveis		
	2. Valor default dos campos		
	3. Barra de rolagem		Verificar a ativação da barra de rolagem em campo somente leitura.
USABILIDADE	VALIDAÇÃO		
	1. Validação da tela e dos seus componentes		
	2. Texto de Ajuda		
	3. Tecla ENTER		
	4. Tecla TAB		Troca de foco entre os campos
	5. Teclas de Atalho		
	6. Texto Tooltip		Texto exibido ao passar o mouse por um ícone
	7. Links de Navegação		
	8. Barra de Rolagem		
	9. Resolução (800x600 e 1024x768)		
MENSAGEM	1. Sintaxe		
	2. Semântica		
	3. Padrão Visual	Verificar componentes de tela	Verificar componentes de tela
NEGOCIO	VALIDAÇÃO DE FLUXOS		
	1. Fluxo Básico		
	2. Fluxos Alternativos		
	3. Fluxos de Exceção		
	4. Regras de Negócio		
	OPÇÕES DE PREENCHIMENTO DE CAMPOS		
	1.Todos os campos obrigatórios vazios		
	2.Todos os campos obrigatórios preenchidos		
	3. Primeiro e Último campos obriigatórios		
	4. Apenas 1 campo obrigatório preenchido		
SEGURANÇA	1. Base de Dados Indisponível		
	2. Sessão Web Expirada		
	3. Permissões de acesso do Usuário		
	4. Garantir que o sistema não preserva dados na tela após operação	Após incluir um registro	
	5. Testes de F5/Refresh		Não realizar nenhuma ação ao apertar F5
	6. Testes de SQL Injection		Testar a inserção de comando SQl no sistema

Figura 3: Guia de Cenários de Teste

5.3 Execução

Nesta fase, os testes serão executados pelo testador. A planilha de execução já está pronta, ela contém os cenários básicos de testes que serão utilizados para testar o sistema. O testador continua tendo toda a liberdade para elaborar e executar outros testes durante a exploração. Para os problemas detectados são abertos registros de solicitação de mudança.

Nem sempre todos os cenários podem ser executados, por questões de prazo, recurso. Algumas diretrizes podem ajudar a priorizar os cenários a serem executados:

Funcionalidades Críticas

Aconselha-se testar as funcionalidades que são mais críticas para o sistema.

Testar Cenários que já apresentaram erros

Já que não dá para testar todos os cenários, testam-se os cenários que já apresentaram erros. A probabilidade de detectar problemas em cenários que já apresentaram erro é maior do que em cenários que não apresentaram ainda.

Garantir Aspectos vitais do Sistema

Devem-se testar os aspectos vitais do software. Os aspectos secundários são menos importantes. A definição de qual aspecto é mais ou menos importante depende das características do sistema e dos riscos envolvidos nele [4].

Dependendo do sistema os aspectos mais importantes podem ser: funcional, segurança e desempenho. Em outro sistema podem ser: funcional, e usabilidade.

Planilha de Execução

A planilha de execução apresentada neste trabalho é uma instância do Guia de Cenário de Teste proposto na seção anterior.

A planilha contém os seguintes campos:

Sistema: O campo deve ser preenchido com o nome do sistema que será testado.

Versão: Este campo é preenchido com a versão da aplicação que está sendo testada. Com base nesse campo pode-se identificar em qual versão do sistema os erros relatados na planilha ocorreram.

Tela: Neste campo informa-se qual tela do sistema está sendo testada.

Executor: Campo preenchido com o nome de quem está executando os testes.

Data: Campo preenchido com a data de execução dos testes.

Os campos de área resultado informam o total de cenários de testes executados, quantos cenários de testes passaram e quantos falharam. Nessa área também existe o campo "tempo total" que deve ser preenchido com a duração da execução dos testes.

Para cada cenário de teste há os seguintes campos:

Resultado: Informa o resultado do teste. Se ele passou, falhou ou está bloqueado.

ID e Resumo: Se o teste falhar, os campos ID e Resumo devem ser preenchidos com essas informações do registro de solicitação de mudança aberto para registrar o problema encontrado.

Observações: Campo que só deve ser preenchido se for necessário guardar alguma informação importante sobre o teste.

A planilha pode possuir várias abas, cada aba corresponde a uma tela do sistema.

A figura 4 mostra a planilha de execução.

Planilha de Execução

Sistema	Versão:	Tela:	Executor:	Data:

Resultado

Passados:	*Cenário(s)*	*Falhos:*	*Cenário(s)*	*Total:*	*Cenário(s)*	*Tempo Total:*

	Cenários	Exemplo	Resultado	Solicitação de Mudança ID	Solicitação de Mudança Resumo	Observações
CAMPO	VALIDAÇÃO DO PREENCHIMENTO DO CAMPO					
	1. Valor maior que valor limite					
	2.. Valor igual ao valor limite					
	3 Valor menor que valor limite					
	4. Valor igual ao valor minimo					
	5. Valor menor que valor minimo					
	6. Em branco (não preenchido)					
	7. Valor maior que valor limite (CtrlC+CtrlV)					
	8. Espaçamento entre caracteres					
	VALIDAÇÃO DO CAMPO DO TIPO NÚMERO					
	1. Caracteres especias	#$%¨&				
	2. Números negativos	-6				
	3. Números decimais	6,78				
	4. Caracteres alfanuméricos	5fg7				
	VALIDAÇÃO DO CAMPO DO TIPO DATA					
	1. Mês inexistente	18/00/2007				
	2. Dia inexistente	32/01/2007				
	3. Dia negativo	-1/01/2007				
	4. Mês negativo	18/-1/2007				
	5. Ano negativo	18/01/-2007				
	6. Caracteres alfanuméricos	dd/mm/2007				
	7. Formato inválido	18/1/2007				
	8. Data maior que atual	data de amanhã				
	9. Data menor que a atual	data de ontem				
	10. Data final menor que a data inicial					
	11. Ano bissexto	29/2/2008				
	VALIDAÇÃO DO CAMPO DO TIPO HORA	Formato HH:MM				
	1. Hora inválida	25:25				
	2. Minuto inválido	12:67				
	3. Hora negativa	-6:57				
	4. Minuto negativo	13:-8				
	5. Hora vazia	:34				
	6. Minuto vazio	15:				
	7. Formato	12:34:23				
	VALIDAÇÃO DE OUTROS CAMPOS					
	1. Validação do campo email	exemplo#exemplo.com				
	2. Validação campo CPF	046.567.345-78				
	VALIDAÇÃO DO CAMPO (OUTROS)					
	1. Somente leitura e/ou editáveis					
	2. Valor default dos campos					
	3. Barra de rolagem					
USABILIDADE	VALIDAÇÃO					
	1. Validação visual da tela e dos componentes					
	2. Texto de Ajuda					
	3. Tecla ENTER					
	4. Tecla TAB					
	5. Teclas de Atalho					
	6. Texto Tooltip					
	7. Links de Navegação					
	8. Barra de Rolagem					
	9. Resolução (800x600 e 1024x768)					
MENSAGEM	1. Sintaxe					
	2. Semântica					
	3. Padrão Visual					
NEGOCIO	VALIDAÇÃO DE FLUXOS					
	1. Fluxo Básico					
	2. Fluxos Alternativos					
	3. Fluxos de Exceção					
	4. Regras de Negócio					
	PREENCHIMENTO DE CAMPOS PARA BUSCA					
	1.Todos os campos obrigatórios vazios					
	2.Todos os campos obrigatórios preenchidos					
	3. Primeiro e Último campos obrigatórios					
	4. Apenas 1 campo obrigatório preenchido					
SEGURANÇA	1. Base de Dados Indisponível					
	2. Sessão Web Expirada					
	3. Permissões de acesso do Usuário					
	4. Garantir que o sistema não preserva dados na tela após operação					
	5. Testes de F5/Refresh					
	6. Testes de SQL Injection					

Figura 4: Planilha de Execução

5.4 Análise dos Resultados

Neste momento são analisados os resultados obtidos com a realização dos testes. Um relatório é elaborado para consolidar os resultados obtidos. São sugeridas as seguintes seções para o relatório:

Cenários dos Testes

Nesta seção são expostos os testes que foram executados para cada funcionalidade, os cenários que foram usados.

Resultados

Informa o status atual do sistema. Através de gráficos são demonstrados os resultados obtidos com a execução dos testes. Para facilitar a visualização dos resultados é indicada a presença de gráficos que mostrem: o percentual de cenários passados, bloqueados e falhos; o percentual de cenários passados por tipo de teste (campo, mensagem, usabilidade, negócio e segurança); o percentual de cenários bloqueados por tipo de teste; o percentual de cenários falhos por tipo de teste; o percentual de registros de defeitos pequenos, médios e grandes.

Registros de Defeitos

Todos os registros de defeitos são listados nesta seção. O identificador, a gravidade e a descrição dos defeitos são informados.

5.5 Considerações Finais

Este capítulo propôs e detalhou cada fase da metodologia para testes exploratórios sugerida. Apresentou um guia de cenários de testes de sistema que pode ser utilizado para várias funcionalidades de vários projetos. E com base nesse guia montou uma planilha de execução que contém informações relevantes para analisar o *status* e acompanhar a evolução de um sistema. Além de fornecer dados para estimativas de esforço.

O capítulo seguinte relatará um estudo de caso da aplicação de uma metodologia semelhante à sugerida em uma empresa.

6. Estudo de Caso

Em uma empresa foi aplicada uma metodologia para testes exploratórios semelhante à exposta neste trabalho. Por questões de confidencialidade a metodologia aplicada na empresa e os artefatos gerados não poderão ser expostos. Neste capítulo serão descritos o ambiente do estudo de caso, a aplicação da metodologia, benefícios para a organização, análise crítica da metodologia pelos usuários.

6.1 Ambiente do Estudo de Caso

A organização onde a metodologia foi aplicada é uma empresa nacional de Tecnologia da Informação. Os processos possuem certificado ISO 9001:2000 e o processo de desenvolvimento é compatível com o CMMI Nível 2.

O processo de teste é maduro e é independente do processo de desenvolvimento. Ele ocorre em paralelo ao processo de desenvolvimento. O processo de teste é seguido por uma equipe de teste independente. Essa é uma equipe qualificada formada por pessoas independentes dos projetos, que contêm uma maior visibilidade a respeito da qualidade do produto final e que têm como finalidade exclusiva testar os sistemas.

6.2 Aplicação da Metodologia

As fases iniciais de planejamento e elaboração dos cenários básicos foram realizadas pela equipe de testes.

A equipe de desenvolvimento realizou a fase seguinte, execução dos testes. Para poder executar os testes, os desenvolvedores receberam um treinamento ministrado pela equipe de testes sobre testes exploratórios e como utilizar a planilha de execução.

A análise dos resultados foi realizada pela equipe de testes.

Os testes exploratórios foram realizados em dois projetos, projeto A e projeto B.

Os projetos A e B são projetos de grande porte e aplicativos Web.

No projeto A foram testadas 15 funcionalidades. Para cada funcionalidade foram desenvolvidos cenários para testes de campo, mensagem, usabilidade, negócio, segurança e navegação.

Três desenvolvedores realizaram os testes no projeto.

No projeto B apenas uma funcionalidade foi testada com testes exploratórios. Foram realizados três ciclos de execução para testar essa funcionalidade. Para a funcionalidade também foram desenvolvidos cenários para testes de campo, mensagem, usabilidade, negócio, segurança e navegação. Os testes foram realizados por dois desenvolvedores.

No projeto A foram relatados noventa registros de defeitos.

No projeto B, no primeiro ciclo de execução foram detectados sete erros, no segundo ciclo de execução foram encontrados cinco erros e na terceira execução nenhum registro de defeito foi aberto.

Os registros encontrados foram corrigidos pelos desenvolvedores e re-testados pela equipe de testes.

6.3 Benefícios da Metodologia para Organização

Com a utilização dessa metodologia, projetos que não estão aderentes ao processo de desenvolvimento, como é o caso do projeto A, podem ser testados utilizando poucos recursos e trazendo resultados rápidos. Uma vantagem é que quem testa o sistema, os desenvolvedores, possui um bom conhecimento do sistema, das regras de negócio.

Um outro benefício é que projetos aderentes ao processo de desenvolvimento, como é o caso do projeto B, mas que não possuem tempo e recursos disponíveis para a realização de testes sistemáticos em todas as funcionalidades podem ter suas funcionalidades testadas dentro do tempo e com os recursos disponíveis.

Os testes viciados realizados pelos desenvolvedores são minimizados com o treinamento sobre testes exploratórios, a utilização da planilha de execução com os cenários básicos a serem testados garante que os erros básicos serão encontrados e direcionam os desenvolvedores na hora de executar os testes.

6.4 *Análise Crítica da Metodologia pelos Usuários*

Após a realização dos testes, os desenvolvedores foram entrevistados sobre a metodologia.

A maioria dos desenvolvedores não teve dificuldades para executar os testes. Um dos pontos positivos levantados por eles foi o treinamento ministrado pela equipe de teste, que fez com que eles entendessem melhor a atividade de teste, como testar, a importância de testar.

A utilização do guia também foi vista como uma boa prática, uma vez que direciona os testes e ajudar a criar outros cenários de teste.

Nenhuma melhoria na metodologia foi sugerida.

6.5 *Considerações Finais*

A utilização da metodologia é bem vista na organização, inclusive pelos desenvolvedores. Apesar da metodologia ter sido aplicado em uma empresa que possui um processo de teste maduro não significa que ela não possa ser implantada em empresas que não possuem um processo de teste ou um processo de desenvolvimento. A metodologia é facilmente adaptada para ser seguida por pessoas que não possuem o perfil de testador.

7. Conclusão

Neste capítulo são apresentadas as considerações finas e os trabalhos futuros.

7.1 Considerações Finais

Como pôde ser visto neste trabalho, mesmo sem muitos recursos de tempo e dinheiro podem ser aplicadas metodologias de teste que garantem uma melhor qualidade do *software* desenvolvido. Como a metodologia apresentada.

Este trabalho de graduação abordou a importância e os ganhos obtidos quando os produtos são testados antes de chegarem aos clientes. Vários conceitos e informações relevantes sobre teste de software foram apresentados. O tema Testes Exploratórios foi bem detalhado expondo características, vantagens, desvantagens e alguns cenários onde eles podem ser usados. Boas práticas para a realização de testes exploratórios foram explanadas, práticas que auxiliam a exploração. Uma metodologia que planeja e orienta a execução dos testes foi desenvolvida baseada nas boas práticas apresentadas. A metodologia planeja e organiza as atividades de teste, define objetivos, ajuda na exploração, minimiza as desvantagens encontradas ao se empregar esta técnica e aumenta a probabilidade detecção de erros. Um guia de cenários de testes que abrange testes de campo, mensagem, negócio, usabilidade e segurança foi construído. O guia desenvolvido é bem genérico e pode ser empregado em diversos projetos. Uma planilha de execução que é baseada no guia de cenários de testes foi concebida. A planilha fornece informações básicas para medição e análise dos testes, acompanhar a evolução do sistema, realizar estimativas de esforço.

As boas práticas apresentadas e a metodologia sugerida podem ser adaptadas e empregadas em qualquer organização. Ela torna as atividades de testes exploratórios menos informais e dependentes de quem executa os testes, garantindo uma cobertura mínima nos testes, e ainda preservando as vantagens da utilização da técnica: resultados rápidos, necessidade de pouco tempo e pouco recurso.

7.2 Trabalhos Futuros

A continuação deste trabalho é adequar a metodologia ao Modelo de Maturidade de Testes (TMM – Testing Maturity Model). Essa adequação visa facilitar a avaliação e a implantação de melhorias para as atividades de teste.

8. Referências

[1] RIOS, Emerson; MOREIRA, Trayahú. Teste De Software.

[2] JONES, T.Capers. Patterns of Software System Failure and Success.

[3] VIANA, Virginia. Um Método para Seleção de Testes de Regressão para Automação, 2006.

[4] Wikipedia, http://pt.wikipedia.org/wiki/Teste_de_software. Acessado em julho de 2007.

[5] BARTIE, Alexandre. Garantia da Qualidade de Software, 2002.

[6] Software QA/Test Resource Center, http://www.softwareqatest.com. Acessado em agosto de 2007.

[7] MYERS, Glenford. The Art of Software Testing.

[8] MOLINARI, Leonardo. Teste de Software: Produzindo Sistemas Melhores e Mais Confiáveis, 2006.

[9] What Is Exploratory Testing? And How It Differs from Scripted Testing. James Back. Disponível em http://www.satisfice.com/articles/what_is_et.shtml.

[10] Exploratory Testing Explained. James Back. Disponível em http://www.satisfice.com/articles/et-article.pdf

[11] BASTOS, Anderson; RIOS, Emerson; CRISTALLI, Ricardo; MOREIRA, Trayahú. Base de Conhecimento em Testes de Software, 2006.