Boas Práticas Adotadas em um Projeto de *Design* de Testes – Um relato de experiência

Polyana Lima Olegário
Centro de Estudos e Sistemas Avançados do Recife - C.E.S.A.R.
Av. Barbosa Lima 81, Bairro do Recife
Cep 50.030-220 – Recife – PE – Brasil
55 81 31345682
polyana.lima@cesar.org.br

Liane Ribeiro Pinto Bandeira
Centro de Estudos e Sistemas Avançados do Recife - C.E.S.A.R.
Av. Barbosa Lima 81, Bairro do Recife
Cep 50.030-220 – Recife – PE – Brasil
55 81 31345637
liane.bandeira@cesar.org.br

ABSTRACT
Well designed test cases allow failures to be found early, hence decreasing the cost to correct that defect, because of this the test design activity has received great attention in the software testing process.

This paper was developed with the goal to improve the test case design process through good practices adopted during the development and maintenance of test suites. With the improvement of the test cases quality, some benefits were brought along, such as improvement on test execution, a decrease in the need to update the tests, cost reduction due to less execution effort or even due to the increase in the probability to find bugs earlier, reduction in the time spent in the testing process overall, amongst others that will be cited in this paper.

RESUMO
Casos de teste bem projetados permitem que falhas sejam encontradas antecipadamente, reduzindo assim, o custo para correção de tal defeito, por este motivo a atividade de *design* de testes tem tido grande destaque em processos de teste de *software*.

Este trabalho foi desenvolvido com o objetivo de aprimorar o processo de *design* de casos de teste através de boas práticas adotadas durante a construção e manutenção das suítes de teste. Com a melhoria dos casos de teste foram trazidos alguns benefícios, tais como melhoria na execução dos testes, redução da necessidade de atualização dos testes, redução dos custos, por meio da diminuição do esforço de execução dos testes ou mesmo através do aumento da probabilidade de se encontrar *bugs* antecipadamente, redução do tempo gasto no processo de teste de forma geral, dentre outros que serão citados neste trabalho.

Categorias e Descrição do Assunto
D.2.9 [**Software Engineering**]: *Testing and Debugging – Tracing*.

Termos Gerais
Design, Desempenho, Experimentos, Verificação.

Palavras-Chave
Design de casos de teste, boas práticas, qualidade do produto, manutenção de suítes de teste e testes de integração

1. INTRODUÇÃO
O processo de teste de *software* tem tido grande importância dentro da área de engenharia de *software*, pois diminui os custos de manutenção e permite que problemas futuros sejam evitados antecipadamente [9]. Atividades de teste de *software* podem ser utilizadas no projeto de qualquer tipo de aplicação para assegurar acurácia, confiabilidade, segurança do produto, dentre outros quesitos de qualidade. Esta atividade é realizada por meio de verificação, cujo objetivo é observar as diferenças entre o comportamento esperado da aplicação e o seu comportamento real [10]. Testes de *software* são também capazes de detectar problemas que eventualmente não tenham sido previstos na especificação do projeto.

Apesar de suas diversas vantagens existem algumas dificuldades em se testar *software*, como a falta de profissionais especializados na área de teste, o desconhecimento de um procedimento de teste adequado, o custo da atividade de teste, dentre outras [7]. Portanto, todo o processo de teste deve ser bem planejado, a fim de obter o máximo de produtividade desejada.

A atividade de teste de *software* é composta por diversos passos, dentre eles: planejamento e controle, análise e *design*, implementação e execução, avaliação do critério de saída e reportagem, atividade de fechamento do teste [5]. A Norma IEEE 829-1998 [6] descreve um conjunto de 8 documentos para auxiliar a cobertura das tarefas de planejamento, especificação e registro das atividades de teste de um produto de *software*. Um dos documentos da especificação dos testes é o documento de Especificação de caso de teste, o qual deve definir os casos de teste, incluindo dados de entrada, resultados esperados, ações e condições gerais para a execução do teste. Tal documento faz referência à atividade de *design* de casos de teste, na qual os casos de teste são construídos ou modificados de acordo com as especificações do produto.

De acordo com Pressman [9], os casos de teste[1] são desenhados para encontrar o maior número possível de erros com o mínimo esforço e tempo possível, portanto essa atividade de construção de casos de teste é bastante importante e desafiadora. Myers [8]

[1] Casos de teste são criados baseados em cenários e cenário é uma seqüência de passos que descreve as interações entre o ator e o sistema [3].

destaca ainda que o projeto de casos de teste é importante, porque testar completamente é impossível - o teste de qualquer programa deve ser necessariamente incompleto -, então se deve tentar desenvolver testes tão completos quanto possíveis.

Diante da importância da eficiência dos casos de testes na atividade de teste de *software*, este trabalho mostra algumas práticas implementadas para melhoria de um processo de teste. Será relatada a experiência de uma equipe de um projeto de *design* de casos de teste durante a implantação de tais práticas. A seção 2 contextualiza o trabalho realizado, a seção 3 apresenta as boas práticas adotadas no projeto assim como os benefícios advindos destas e, por fim, a seção 4 terce algumas considerações finais sobre este trabalho realizado.

2. CONTEXTUALIZAÇÃO

O Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R.) [1], situado em Recife – Pernambuco foi o local aonde este trabalho foi desenvolvido e consiste numa organização sem fins lucrativos que tem como missão a transferência auto sustentada de tecnologia entre a universidade e a sociedade. O C.E.S.A.R foi reconhecido como nível 2 do SW-CMM (*Capability Maturity Model for Software*) [2] em 2003 em uma área de projetos que desenvolve *software* para o mercado de telefonia celular e em 2007 almeja ser avaliado oficialmente como nível 3 do CMMI.

Existem diversos projetos ligados ao C.E.S.A.R e um deles é o programa de testes de aplicativos em telefonia móvel, pertencente à Motorola Brasil. Este programa é denominado Brasil Test Center (BTC) e engloba, além do C.E.S.A.R, organizações como o Instituto Eldorado em Campinas/SP, o Centro de Informática da Universidade Federal de Pernambuco (CIn) e a Universidade Federal de Santa Catarina (UFSC).

A experiência aqui relatada foi extraída de um dos projetos do programa BTC, o projeto de *Test Design*, o qual é responsável por construir casos de teste para que a equipe de execução de testes, ou testadores, execute os testes de forma manual ou automática. Este projeto, assim como vários outros dentro do programa BTC, possui um processo bem definido.

Ao longo do processo de *design* de teste, vários casos de testes vêm sendo construídos e agrupados em suítes, tais suítes foram aproveitadas de uma versão do produto para outra. Depois de varias alterações nos testes para adaptá-los ou até mesmo para adicionar cobertura de novas funcionalidades, sentiu-se dificuldade em controlar a manutenção destas suítes, além disto, a equipe de testadores passou a reportar muitos defeitos nos casos de teste. A partir destes fatos, houve a necessidade de melhorar a qualidade dos casos de teste, conseqüentemente melhorando a qualidade de execução dos testes, a qualidade do produto e reduzindo o esforço e custo desta etapa.

3. BOAS PRÁTICAS ADOTADAS NO PROJETO

Durante a atividade de escrita ou correção de casos de teste existem fatores que podem contribuir para que o caso de teste seja capaz de encontrar falhas[2] no sistema e que seja de fácil compreensão para os testadores. Este último fator é importante para evitar que o caso de teste seja executado de forma ambígua.

A fim de garantir a melhoria da qualidade do produto através de casos de teste bem escritos e projetados, algumas práticas foram adotadas pelo projeto de *design* de casos de teste do CESAR em conjunto com a Motorola, as quais serão detalhadas a seguir.

3.1 Sessões de *Brainstorm*

Durante o *design* de um novo caso de teste baseado numa nova funcionalidade desenvolvida, é preciso definir quais aplicações, que fazem interação com a nova funcionalidade, farão parte do caso de teste. Sabemos que é inviável testar todas as interações possíveis [5, 7] e que alguns requisitos podem não ser testados neste nível de teste. Segundo Craig [4], a idéia de uma sessão de *brainstorm* é criar listas de idéias para ser testadas, por isso utilizamos de sessões de *brainstorm*, as quais consistem de reuniões entre todos os membros da equipe de *design* de teste para trocar idéias, analisar as sugestões propostas e propor cenários de casos de teste.

Durante a sessão de *brainstorm* a nova funcionalidade é apresentada pelo *designer* dos casos de teste e os outros membros sugerem interações com tal funcionalidade, baseadas na experiência adquirida durante a atividade de construção de casos de teste. A intenção destas sessões é fazer com que toda a equipe compartilhe o conhecimento adquirido e, principalmente, que os casos de testes sejam construídos de maneira que atinjam o máximo de cobertura possível e ao mesmo tempo encontrem o maior número de falhas na nova funcionalidade.

As sessões de *brainstorm* podem atingir melhor seu objetivo quando têm participação de engenheiros de requisitos e usuários do sistema. A participação dos engenheiros de requisitos garantiria que os requisitos mais importantes estão sendo testados além de antecipar a validação do documento de requisitos antes da execução dos testes. Já os usuários do sistema seriam capazes de fornecer suas expectativas em relação ao sistema, lembrando que um dos fatores principais do teste de *software* é garantir que o sistema satisfaz as expectativas e necessidades do cliente [5].

A prática de sessões de *brainstorm* permitiu melhor visualização da funcionalidade a ser testada, mesmo não sendo possível incluir engenheiros de requisitos e usuários do sistema nas sessões de *brainstorm*. Também por meio dessa prática os membros da equipe de *design* de testes passaram a ter mais oportunidade de trocar conhecimentos.

3.2 Limitar a Quantidade de Componentes por Caso de Teste

Os casos de teste criados no contexto deste projeto são testes realizados em um ambiente similar ao ambiente do cliente, porém os testes são chamados de testes de integração, por considerar como fator principal a integração entre as diversas funcionalidades do dispositivo testado.

[2] Falha refere-se a algum problema no produto encontrado durante a execução dos casos de teste, quando os resultados obtidos não batem com o resultado esperado, e são detectadas principalmente pela equipe de execução dos testes.

Tais testes podem conter diversos componentes[3] sendo testados simultaneamente, portanto, limitar a quantidade de componentes por caso de teste permite uma melhor avaliação da cobertura da suíte. Observa-se que por meio dessa prática é possível indentificar quais componentes estão mais sujeitos a falhas em uma determinada versão do sistema.

A escolha de quais componentes irão compor um caso de teste de integração depende das iterações que é possível realizar entre o componente principal do teste e outros componentes (que possam interagir com ele). Além disso, cada componente possui diversas funcionalidades, portanto pode haver vários casos de teste com o mesmo componente principal, pois neste caso, tal componente pode interagir com diversos outros componentes.

Antes da adoção desta prática era difícil determinar qual componente falhou num caso de teste, visto que os testes continham diversos componentes sendo testados e os mesmos não eram visivelmente indentificados. Esta prática, portanto, auxiliou a execução dos testes e detecção de falhas por componente, bem como o planejamento da execução dos testes.

A utilização desta prática ainda ajudou os *designers* a focar melhor no objetivo principal do teste, influenciando positivamente os testadores, os quais puderam se especializar em determinados componentes do sistema.

3.3 Escrever Casos de Teste em Alto Nível

Outra medida necessária para evitar alteração nos casos de teste sempre que havia modificação de interface do produto, foi escrever os casos de teste em linguagem de alto nível. Desta forma foi possível manter os testes objetivos, claros e ao mesmo tempo livres de detalhes excessivos. Medidas como evitar a verificação de palavras ou frases advindas da interface do sistema sendo testado, como por exemplo mensagens de confirmação ou erro, nomes de botões ou telas, a não ser quando extremamente necessário; evitar usar passo a passo detalhados sem objetivo de verificação, neste caso foi mais eficiente agrupar passos sem resultados desejados em um único passo com um objetivo coerente com o objetivo do caso de teste.

As medidas anteriormente descritas tornaram os testes mais executáveis (isto é, com menos ocorrência de bloqueio[4]), pois passaram a ser mais independentes da versão do sistema testado. Tal prática é bastante válida principalmente para produtos onde há mudanças na interface ou navegabilidade e cujas funcionalidades básicas continuam inalteradas, permitindo que os casos de teste sejam reutilizados. Além disso, houve diminuição de atualizações dos testes, visto que eles não estavam mais diretamente dependentes da interface da aplicação, a qual era constantemente modificada.

[3] No contexto desse trabalho, um componente reúne um conjunto de funcionalidades que compartinham objetivos e recursos em comum.

[4] Casos de teste bloqueados são testes que estão com algum defeito no próprio caso de teste ou quando, por algum motivo externo, não podem ser executados, como problemas de ambiente. Então estes permanecem bloqueados até a solução do problema.

3.4 Adicionar Toda Informação Necessária ao Caso de Teste

No projeto em questão, a suíte de testes é executada diversas vezes por equipes em diferentes regiões geográficas e até mesmo em diferentes fusos horários. Portanto, os casos de teste precisam conter toda informação necessária para execução do teste, isto é, as configurações iniciais de um teste, as informações sobre modificações no teste, os requisitos associados aos passos do teste, dentre outras informações, de tal forma que o caso de teste seja executado sem necessidade de auxilio do projetista do teste.

Foram melhoradas as condições iniciais dos casos de teste, com o objetivo de agilizar as execuções e diminuir o esforço despendido na busca de informações necessárias para o teste. Para tal, adotou-se que, qualquer informação que antecede o teste, como assessórios necessários, configurações de ambiente e do dispositivo a ser testado, devia ser citada nas condições iniciais do teste . Assim, os testes passaram a ser executados de forma mais eficaz, pois os testadores já disponibilizavam de toda informação requerida para execução do teste antes de iniciá-lo, diminuindo as interrupções que ocorriam durante a execução do teste .

As informações associadas à mudança do caso de teste foram sempre mantidas atualizadas, como a data das modificações e identificação de quem modificou o teste. Desta forma, torna-se mais fácil identificar o autor do caso de teste e suas mudanças. Isso auxilia o testador a retirar possíveis dúvidas sobre o caso de teste.

A iniciativa de rastrear cada passo de um caso de teste a um requisito foi tomada para facilitar a atividade de testadores e *designers* de testes, pois quando uma falha é encontrada por um testador, este irá facilmente identificar o requisito associado à falha, assegurando que se trata de um problema no sistema. Para os *designers*, quando um requisito é modificado, este poderá ser rapidamente encontrado na suíte de testes e o passo do caso de teste será atualizado antecipadamente.

Estas iniciativas facilitaram a execução e manutenção dos casos de teste, consequentemente, esforços antes desperdiçados na identificação da falha ou do caso de teste associado ao requisito modificado foram amenizados.

3.5 Refletir Cenários Comumente Utilizados pelo Usuário Final

Casos de teste com passos mais próximos do comportamento do usuário final possuem mais chances de encontrar falhas com severidade mais alta, ou seja, falhas que são realmente críticas ao usuário. O objetivo do projeto de teste é encontrar a maior quantidade possível de falhas, com maior severidade possível, num menor intervalo de tempo. Diante deste fato, a iniciativa foi atribuir uma classificação ou peso para cada caso de teste, quanto mais usual o teste, mais próximo do usuário final e dos objetivos do cliente, maior a classificação do caso de teste.

Desta forma, ao se construir novos casos de teste, o *designer* deve ter em mente um cenário, o qual possa ter a classificação maior possível para a funcionalidade que será verificada. Conseqüentemente, se alguma falha for encontrada em um caso de teste que possui alta classificação, a falha poderá ter maior severidade.

A prática de construir casos de teste próximo ao comportamento do usuário permite que falhas mais graves, ou com maior severidade, sejam encontradas, além de manter o foco do teste no cliente.

3.6 Evitar Casos de Teste Exaustivos

Antes da iniciativa de aplicar estas boas práticas na suíte de testes, existiam alguns casos de teste com mais de 30 passos ou mesmo alguns que demoravam mais de 20 minutos para ser executado, isto tornava o caso de teste exaustivo demais para os testadores e despendiam esforço acima da média.

Testes grandes e que tomam muito tempo para executar tendem a causar dispersão no testador quando executados manualmente e a perder o foco nas funcionalidades testadas. Desta forma, foi definida como uma prática reduzir a quantidade de passos em cada teste, para este caso, mantendo entre seis a quinze passos em cada teste. Os passos foram agrupados por áreas semelhantes em novos casos de teste e funcionalidades obsoletas ou estáveis foram removidas.

A partir destas modificações os casos de testes passaram a manter o foco no objetivo principal de teste sem dispersar o testador e consumindo menos esforço para sua execução.

3.7 Manter o Time de Execução dos Casos de Teste Constantemente Informados

Um dos pontos de melhoria do processo de desenvolvimento de *software* está na comunicação entre os desenvolvedores, *designers*, engenheiros de requisitos e testadores. A melhoria da comunicação entre os envolvidos no projeto uniformiza o entendimento dos objetivos do projeto.

No processo de *design* de casos de teste do projeto em questão está inclusa a atividade de realizar inspeções no caso de teste antes de disponibilizá-lo para execução. Devem participar destas inspeções os testadores do teste, o *designer* do caso de teste, assim como outros membros que detenham informação técnica sobre o teste. Esta medida propõe antecipar possíveis erros no teste, assim como permite que testadores possam compartilhar seu conhecimento sobre determinadas funcionalidades, enriquecendo os casos de teste.

Depois desta prática, a equipe de execução dos casos de teste passou a ter mais envolvimento na construção e alteração dos testes.

3.8 Tornar Casos de Teste aptos para Automação

Alguns casos de teste do projeto possuíam uma configuração de ambiente muito complexa, tomando muito tempo do testador. Em muitos destes casos não era possível diminuir esse esforço para execução manual dos testes, assim, foi incentivado a tornar alguns desses testes aptos a serem automatizados. Isto foi possível através do agrupamento dos passos que necessitavam dessa configuração em poucos testes, diminuindo assim a quantidade de casos de teste com tais configurações. Uma análise desses casos de teste, realizada posteriormente, identificou as funcionalidades já suportadas pelo ambiente de teste automático, para que estes se tornassem casos de teste automáticos.

Outro fator importante para automação de casos de teste foi a necessidade de manter testes que realizam verificações usuais no produto. Muitos desses testes já não encontram tantas falhas quanto no início da execução dos ciclos de testes, porém se tratam de casos de teste que verificam funcionalidades essenciais do produto, e por isso não podem deixar de ser executados. Algumas adaptações nesses casos de testes foram feitas, para que o ambiente de testes automáticos pudesse executá-los.

Essa prática, além de garantir a validação de funcionalidades básicas através da execução automática de alguns casos de teste, dispensou esforço da execução manual e diminuiu o tempo de execução da suíte de testes.

4. CONSIDERAÇÕES FINAIS

Partindo das práticas apresentadas neste projeto, o projeto de casos de teste poderá construir testes de mais fácil compreensão, capazes de encontrar falhas mais severas e que exijam poucas correções durante o ciclo de execução dos testes.

Aplicando-se as boas práticas descritas em conjunto com a experiência do *designer* de casos de testes, é possível obter uma melhoria significativa na execução dos testes, pois os casos de testes tornam-se mais concisos e possuem informações completas para auxiliar a execução pelos testadores.

Obtém-se também uma diminuição do tempo para confirmação da descoberta de uma falha, pois cada passo possui um requisito associado. Além disso, houve melhoria na comunicação entre os *designers* de teste e os testadores, através da participação destes nas inspeções. Isso fez aumentar a qualidade dos testes, visto que se agregou a prática da execução dos testadores com o conhecimento técnico dos *designers*, nas mais diversas funcionalidades do sistema.

O esforço para correção de casos de teste também pôde ser minimizado, através das práticas da escrita dos casos de teste em alto nível e da adição das informações necessárias para execução do teste.

As práticas de limitar a quantidade de componentes e de passos por caso de teste permitiu focar mais fortemente no objetivo central do teste, influenciando positivamente a execução dos testes e conseqüentemente a qualidade do produto.

A construção de casos de teste próximo ao comportamento do usuário final, verificando o que é mais utilizado pelo usuário, também permitiu melhorar a qualidade dos casos de teste.

A prática de automação de casos de teste garantiu a execução de testes essenciais ao produto, além de diminuir o esforço necessário para execução da suíte de testes.

Utilizando-se as práticas descritas neste projeto, foi possível obter melhorias no processo de construção de novos casos de teste, na execução dos casos de teste, nos custos com a atividade de teste e principalmente na qualidade do produto testado. Estas práticas facilmente podem ser utilizadas por demais projetos que pretendem construir casos de teste completos e de forma eficiente, servindo como diretriz para projetistas, testadores e gerentes de teste.

5. REFERÊNCIAS

[1] C.E.S.A.R *Centro de Estudos e Sistemas Avançados do Recife,* http://www.cesar.org.br, acessado em 29 de julho de 2007.

[2] CMMI Product Team *CMMI for Development*, Version 1.2 (CMU/SEI-2006-TR-008, ESC-TR-2006-008). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. USA, 2006.

[3] Copeland, L. *A Practitioner's Guide to Software Test Design*, Artech House Publishers, 2004.

[4] Craig, R. D., Jaskiel, S. P. *Systematic Software Testing*, Artech House Publishers, 2002.

[5] Graham, D., Veenendaal, E., Evans, I., Black, R. *Foundations of Software Testing: ISTQB Certification*, Thomson Learning, 2007.

[6] IEEE computer Society; IEEE Std 829: *Standard for Software Test Documentation,* Sep. 1998.

[7] Molinari, L. *Testes de Softwares – Produzindo Sistemas Melhores e Mais Confiáveis*, Ed. Érica, São Paulo, 2003.

[8] Myers, G. J. *The Art of Software Testing*, 2. ed, John Wiley & Sons, Inc., 2004

[9] Pressman, R. S. *Engenharia de Software*. 5 ed., McGraw-Hill, 2002

[10] Sommerville, R. *Engenharia de Software*, 6 ed, Addison Wesley, 2003.