UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

CAMPUS LUIZ MENEGHEL

SAULO DO NASCIMENTO MACHADO

UTILIZAÇÃO DA PRÁTICA DE TDD GERANDO CASOS DE TESTE PARA IMPLEMENTAÇÃO

Bandeirantes

2010

UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

CAMPUS LUIZ MENEGHEL

SAULO DO NASCIMENTO MACHADO

UTILIZAÇÃO DA PRÁTICA DE TDD GERANDO CASOS DE TESTES PARA IMPLEMENTAÇÃO

Bandeirantes

2010

SAULO DO NASCIMENTO MACHADO

UTILIZAÇÃO DA PRÁTICA DE TDD GERANDO CASOS DE TESTES PARA IMPLEMENTAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Norte do Paraná - Campus Luiz Meneghel, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação, orientado pelo Profº Carlos Eduardo Ribeiro.

Bandeirantes

2010

SAULO DO NASCIMENTO MACHADO

UTILIZAÇÃO DA PRÁTICA DE TDD GERANDO CASOS DE TESTES PARA IMPLEMENTAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Norte do Paraná - campus Luiz Meneghel, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação, orientado pelo Profº Carlos Eduardo Ribeiro.

COMISSÃO EXAMINADORA

_____
Prof.º Carlos Eduardo Ribeiro.
Universidade Estadual do Norte do Paraná – *Campus* Luiz Meneghel

_____
Prof.ª Daniela de Freitas G. Trindade.
Universidade Estadual do Norte do Paraná – *Campus* Luiz Meneghel

_____
Prof.º José Reinaldo Merlin.
Universidade Estadual do Norte do Paraná – *Campus* Luiz Meneghel

Bandeirantes, __ de ___ de 2010.

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus e a minha família, meu pai Levi Isaias Machado, minha mãe Jemima Carvalho do Nascimento Machado, meu irmão Abner do Nascimento Machado e minha cunhada Carina Kimura Machado, pelo grande esforço em me manter na faculdade longe de casa, não medindo esforços para que eu pudesse estudar e alcançar tudo aquilo que desejo para minha vida. Também dedico este trabalho a minha namorada e sua família, Dona Salete Siqueleiro Casagrande, Celso Sicheleiro Casgrande e a meu amor Jéssica Casagrande, que souberam em muitos momentos me ter como um membro da família.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ser minha fortaleza nos momentos difíceis e pelas forças dadas durante toda a minha vida acadêmica, sem a qual não seria possível chegar até aqui.

Agradeço a minha família: meu pai Levi Isaias Machado que muitas vezes me chamou para conversar e mesmo sem concordarmos sempre, na maioria, sei que seus conselhos me engrandeceram como pessoa e aumentaram minha admiração por ele; minha mãe Jemima Carvalho do Nascimento Machado que sempre esteve buscando me divertir e todos a minha volta, sabendo no fundo que ela foi à pessoa que mais sofreu com nossa distância, meu irmão Abner do Nascimento Machado pelos momentos de alegria que sempre tivemos, semente pelo fato de estar um perto do outro, minha cunhada Carina Kimura Machado que sempre foi especial para mim e me mostrou que era capaz de fazer meu irmão feliz, como vem fazendo até hoje ainda mais com a presença do meu sobrinho Luquinhas que ainda não conheço mais sei que será como um filho para mim. Sei que com a ajuda deles pude montar meu caráter e com a força dada por eles tive muita garra para alcançar tudo o que desejei, sabendo que eles me apoiarão em todos os momentos de minha vida.

Agradeço grandemente a minha noiva e futura esposa Jéssica Casagrande pela força e apoio dado durante todo o período em que estivemos juntos, aos 3 maravilhosos anos, 3 especiais meses e 28 lindos dias que até hoje me fizeram muito mais feliz, com grande carinho, força e dedicação sem a qual não saberia superar as dificuldades.

Agradeço a todos os professores da UENP – CLM que participaram ativamente da construção de meu conhecimento, me auxiliando direta e indiretamente na realização, não só deste trabalho, mas também na realização de todos os meus trabalhos durante minha vida acadêmica.

Agradeço ao meu professor orientador Carlos Eduardo Ribeiro não só pelas instruções dadas para realização deste trabalho, mas também pela amizade e companheirismo que ele teve comigo, me auxiliando também em minha vida pós-faculdade, me orientando como amigo.

Agradeço em especial aos professores Merlin e Daniela que me deram a oportunidade de participar do projeto “Estruturação e Capacitação na Gestão da

Produção e Comercialização de Uva", o qual interferiu grandemente em meu futuro, pois através deste tive uma melhor formação e pude direcionar meus estudos e meus esforços para vida após a faculdade.

Agradeço a todos de minha turma pelos momentos de alegria, sempre vendo graça nos momentos de dificuldade que passamos ao longo do curso.

Agradeço em especial aos meus amigos que sempre estiveram dispostos a me ajudar durante os quatro anos de curso, dentre eles: aos fieis amigos de morada da Rep. Caverna Lélis e Mateus; aos moradores da Rep. Califórnia, Jaime, Monollo, Jonas, Kichiro, Vitinho e Miltinho; aos moradores da Rep. Talaricos, Bruno, Montanha, Tiago e Gutt; aos membros da equipe LAS, Andrezão, Gordinho, Lokura, João, Matsui, Cowboy, Felipe, Ruan e a eterna integrante do nosso projeto Betinha, seu marido Beto e sua linda filha Giulia.

“Procure ser um homem de valor, em vez de ser um homem de sucesso”.

(Albert Einstein)

RESUMO

Devido à grande necessidade tecnológica encontrada pelas empresas, a utilização de *softwares* torna-se imprescindível em todas as áreas do conhecimento, sendo requisitado que estes softwares possuam o maior nível possível de qualidade. Devido ao curto prazo, às vezes, exigido pelos clientes na entrega do programa em desenvolvimento, se faz necessário agilizar o processo de desenvolvimento dos *softwares.* Este trabalho tem por objetivo avaliar a utilização da técnica de desenvolvimento orientado a testes (TDD), para otimizar e agilizar o processo de desenvolvimento de software. Para isso, será necessário o levantamento e definição dos requisitos juntamente ao cliente, assim como a definição dos modelos de casos de testes a serem utilizados para propiciarem uma otimizada codificação, dessa forma espera-se possibilitar a realização de uma análise dos benefícios e das desvantagens encontradas na utilização do TDD.

Palavras-chave: Teste de Software; Desenvolvimento Orientado a Testes; Casos de Teste; Qualidade de *Software*.

ABSTRACT

Due to the great technological need found by companies, the use of software become essential in all areas of knowledge, being required to possess the highest level of quality. Due to the short term, sometimes required by customers in the delivery of program development, is needed to expedite the process of software development. This study aims to evaluates the use of the technique of test-driven development (TDD) to streamline and expedite the process of software development. This will require the survey and definition of requirements along with the customer as well as the definition of models of test cases to be used to provide an optimized encoding, thus it is expected to enable the realization of an analysis of the benefits and disadvantages found in the use of TDD.

Keywords: Software Testing; Test-Driven Development; Test Cases; Software Quality.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No mundo atual há uma grande evolução tecnológica, na qual está mais do que evidente que toda empresa, de qualquer área do conhecimento, necessita de um sistema informatizado para gerir a informação presente dentro da organização, sendo este conhecimento um dos bens mais valioso para o sucesso da empresa.

Assim sendo, as empresas necessitam cada dia mais de um software em seu ambiente de trabalho. No entanto, empresas voltadas ao ramo de desenvolvimento de software encontram alguns problemas para a confecção de um software que atenda às necessidades de seus clientes, visto que geralmente nem mesmo estes clientes sabem dizer o que necessitam para que o software funcione.

Desta maneira as empresas de desenvolvimento necessitam utilizar uma metodologia para produção do software, a qual propicie que os requisitos do software sejam passados pelo cliente mesmo durante o processo de codificação, visto que após o cliente começar a utilizar parte do software, passa a ser natural que este sinta falta de alguma coisa que o sistema deva executar, assim mudando os requisitos do sistema a ser confeccionado de maneira a atender às suas necessidades.

Para este fim, as empresas podem se beneficiar de uma metodologia ágil para desenvolver seu software. Estas metodologias são caracterizadas, dentre outros aspectos, a trabalharem com requisitos dinâmicos (que podem ser alterados com o passar do tempo de desenvolvimento) ou até mesmo desconhecidos (que ocorrer quando nem mesmo o cliente sabe, no início do projeto, que tal requisito deve existir).

Koscianski e Soares (2007) afirmam que outra característica observada está em disponibilizar, em pouco espaço de tempo, uma nova versão do software para que o cliente possa usar e identificar novas funcionalidades que são requeridas ao sistema.

A utilização destas metodologias propicia que sejam realizadas novas práticas de desenvolvimento, dentre as quais se encontra a prática de desenvolvimento dirigido a testes (*Test-Driven* Development - TDD). Conforme Sengupta (2004) empregar a prática de TDD é uma solução para frequente mudança referente aos requisitos do software, visto que nesta prática são

construídos casos de teste que servirão de base para a realização da codificação das funcionalidades, sendo assim caso seja necessário alterar, excluir ou inserir um novo requisito, deve-se apenas alterar, excluir ou inserir um caso de teste referente à funcionalidade.

Sendo que este caso de teste é composto de um conjunto de entradas que são requeridas para uma funcionalidade, indicando também quais seriam as saídas corretas na efetiva execução da funcionalidade (SENGUPTA, 2004).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

A realização deste trabalho possui o objetivo de analisar a utilização da prática de desenvolvimento dirigido a testes (TDD) para implementação de um software, através da geração de casos de testes que auxiliarão o desenvolvimento do software como um todo.

1.1.2 Objetivos Específicos

Este trabalho possui como objetivos específicos:

- Obtenção de maior conhecimento de como é realizado o desenvolvimento de software através de uma metodologia ágil;
- Elaboração de casos de testes necessários para maior entendimento a respeito das funcionalidades do software a ser implementado;
- Utilização da prática TDD para realizar a implementação do software SISCCOMP;
- Análise de benefícios e desvantagens da utilização da prática TDD;

1.2 Justificativa

Não somente as empresas de desenvolvimento de software, como também as demais empresas de todas as áreas do conhecimento, objetivam entregar para seus clientes um produto de qualidade que disponibilize o maior nível de satisfação possível.

Sendo assim, este trabalho se justifica pela necessidade existente nas empresas de software de trabalhar com uma metodologia de desenvolvimento que possibilite: ao cliente alterar os requisitos de seu produto em qualquer fase do desenvolvimento, sem que isto acarrete em uma grande elevação no custo; geração de um produto somente com aquilo que foi pedido e em pouco tempo; além da obtenção de bons resultados no âmbito da geração de um produto com menor número de defeitos.

1.3 Metodologia

Devido à necessidade de se obter maior conhecimento a respeito de alguns assuntos, que foram abordados nos objetivos do trabalho, pode-se classificar a pesquisa como sendo exploratória.

Esta pesquisa se caracteriza também como um estudo empírico, visto que através da experiência se deve obter um conhecimento maior a respeito do tema e pretende-se analisar a hipótese de que com a utilização da prática de TDD pode-se implementar uma funcionalidade como maior qualidade e de maneira mais satisfatória para o cliente.

1.4 Estrutura da Pesquisa

O trabalho segue estruturado da seguinte forma: na Seção 2 é apresentado a fundamentação teórica que aborda a prática de teste de software, assim como sua importância e objetivos; na Subseção 2.1, apresentam-se os testes utilizados para completar o sistema, nas Subseções 2.1.1 e 2.1.2, apresentam os testes realizados

no software completo, nas Subseções 2.1.3, 2.1.4, 2.1.5 e 2.1.6 são apresentados conceitos referentes à teste de software.

Dessa forma, a fundamentação segue abordando a necessidade de uma boa qualidade de software e os problemas apresentados na utilização do teste de software convencional, isto nas Subseções 2.2 e 2.3 respectivamente.

A Subseção 2.4 apresenta a prática de *Test-Driven Development* (TDD), sendo sua utilização objetivo deste trabalho. Na Subseção 2.4.1 segue-se com a fundamentação sobre metodologias ágeis, que contextualiza o tema objetivo do trabalho, detalhando na Subseção 2.4.1.1 a metodologia *Scrum*, a qual será empregada a prática de TDD a fim de analisar as vantagens e desvantagens. Sendo que o cronograma do trabalho é apresentado na Seção 3.

Por fim, a Seção 4 e 5 apresentam respectivamente os resultados e as conclusões incluindo um direcionamento de trabalhos futuros potenciais.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Teste de Software

Afirma Barbosa (2007) que o desenvolvimento de software é composto de vários processos e atividades, os quais mesmo utilizando técnicas adequadas, métodos eficientes para o desenvolvimento e ferramentas bem escolhidas para isto, no final é bem provável que alguns erros venham a ocorrer no produto.

Caso não se identifique os erros antes da entrega do produto ao cliente, estes provocarão insatisfação por parte do cliente, sendo isso catastrófico para as empresas de software que objetivam total satisfação do cliente. Visto também que estas empresas destinam investimentos e incentivos consideráveis para a qualidade do software, objetivando obter uma certificação reconhecida como as que são conferidas pela *International Organization for Standardization* (ISO), por exemplo ISO 9001, ISO 14000 entre outras (FRANZEN; BELLINI, 2005).

Conforme Pressman (2002) os erros de um software são tão diversificados que podem vir a ocorrer até mesmo no início do processo de desenvolvimento, sendo que isso geralmente ocorre quando as funcionalidades e os objetivos do produto são errôneos ou foram mal especificados pelo cliente durante o levantamento de requisitos do produto, sendo esta a fase na qual, em geral, os erros ocorrerem com mais freqüência.

O autor afirma ainda que em geral, os erros e defeitos são causados por falta de habilidade dos membros da equipe de desenvolvimento, durante a realização dos processos de desenvolvimento de software e até mesmo por falta de qualidade na comunicação entre a equipe, sendo assim imprescindível a garantia da qualidade durante todo o projeto.

Desta maneira, visando uma maior garantia na qualidade do software tem-se a utilização das atividades de VV&T – Verificação, Validação e Teste, que objetivam a identificação de erros decorrentes ao projeto de software e os riscos que estão associados ao software (BARBOSA, 2007).

Dentre as atividades de garantia da qualidade, tem-se que o teste é a atividade mais utilizada, pois através dele tanto a empresa de desenvolvimento quanto os clientes obterão uma confiabilidade maior em relação ao software, visto

que esta atividade é uma análise dinâmica, na qual serão identificados e solucionados erros que persistirem no software (BARBOSA, 2007).

Segundo Pressman (2002) os testes são projetados com o objetivo de encontrar a maior quantidade de erros e defeito utilizando a quantidade mínima de tempo e esforços da equipe de desenvolvimento. Outro ponto a ser levado em consideração é que os testes efetuados com sucesso descobrirão erros que ainda não foram descobertos em relação ao programa.

Desta maneira os testes deveram constatar que o software está sendo executado de maneira a atender as especificações do cliente e dar-lhe satisfação em relação à utilização do software. Porém, é importante salientar que os testes não demonstram a ausência de alguns erros no software, mas sim a presença de erros e defeitos, para que eles sejam solucionados antes da entrega do produto ao cliente (PRESSMAN, 2002).

Torres-Zenteno (2007) argumenta que a atividade de teste pode gerar uma confusão no entendimento de pessoas que não possuem contato frequente com a atividade, visto que o foco principal dela está no objetivo de garantir o maior nível possível de qualidade para o software em desenvolvimento, porém a atividade é considerada bem realizada quando identifica o maior número possível de erros.

Nita (2007) alerta sobre a importância da segregação, segundo o autor é importante atentar-se também que a realização da atividade de teste deve ser realizada por uma equipe composta por membros diferentes da equipe de desenvolvedores que programaram as funcionalidades, isto para que a lógica de programação não vicie o desenvolvedor dos testes. Sendo que outro motivo relevante para realização ser feita por uma equipe independente, está na importância e seriedade desta atividade, que deve ser planejada para execução, assim não ocasionando vício de verificação da aplicação, na qual situações relevantes poderiam não ser levadas em consideração, assim não sendo tão efetiva a identificação e eliminação de falhas decorrentes da especificação ou implementação.

A definição de Neto (2007) a respeito da atividade de teste é que ela é aplicada no produto final e completo, devendo indicar se o produto atingiu suas especificações e funcionou de maneira aceitável ao ambiente para o qual foi projetado, assim gerando satisfação para o cliente.

Porém, baseando-se no conceito de Sommerville (2003), para que o

software seja considerado completo, a atividade de teste deve ser dividida especificamente em duas fases que objetivarão a obtenção de um sistema completo: o teste estrutural e o teste comportamental.

Como afirmado por Maldonado (2004), o teste comportamental é conhecido também como teste de caixa-preta pelo fato de definir o software como uma caixa fechada, na qual somente pode ser avaliado a parte externa, observando quais são os dados de entrada inseridos pelos usuários e quais seriam as saídas pertinentes ao dados de entrada. Desta maneira, serão consideradas e avaliadas somente as funções do software, sem considerar os detalhes do código implementado.

Conforme afirma Maldonado (2004), o teste estrutural é conhecido também como teste de caixa-branca, este tipo de teste objetiva avaliar a estrutura do software, observando detalhadamente a estrutura interna da implementação, verificando “linha por linha” o código elaborado, para realização de uma análise mais específica do software.

Nos tópicos a seguir serão apresentados alguns tipos de teste de software, os quais integrados e realizados objetivam garantir a completeza do desenvolvimento do software em relação à implementação de um produto com qualidade, entre eles estão: teste de unidade, teste de integração, teste funcional, teste de desempenho, teste de aceitação e teste de integração (PFLEEGER, 2004).

2.1.1 Teste de Unidade

Esta fase ocupa-se de verificar o real funcionamento de cada unidade do software, averiguando se as funções especificas de cada módulo estão sendo implementadas de maneira correta (SOMMERVILLE, 2003).

2.1.2 Teste de Integração

Nesta fase as unidades são integradas e formam sistemas parciais ou sistemas completos, estes testes deveram identificar se algum problema é ocasionado pela integração de vários módulos, sendo possível realizar o teste das funcionalidades e do desempenho do software como um todo (SOMMERVILLE,

2003);

Segundo Pfleeger (2004), após o sistema ser definido como “completo”, existem outros tipos de testes a serem realizado no sistema como um todo, tais como: teste funcional, teste de desempenho, teste de aceitação e teste de instalação.

2.1.3 Teste Funcional

Após ter a segurança de que as informações estão sendo trocadas entre as diversas unidades, como foi projetado, é necessário averiguar se o sistema possui a funcionalidade requerida. Assim esta atividade assegurará que o sistema execute todas as funções que foram definidas no levantamento de requisitos, desta maneira garantirá que o sistema está funcionando (PFLEEGER, 2004).

Deve-se assim identificar que os requisitos são documentados de duas maneiras: primeiramente da maneira que são levantados perante o cliente, identificando o que ele deseja que o sistema realize quando pronto, a segunda maneira de documentação possui os requisitos levantados pelo cliente juntamente com o conjunto de softwares e hardwares que serão utilizados pelos desenvolvedores para elaboração do software, sendo que esta última documentação é a verificada no teste funcional (PFLEEGER, 2004).

2.1.4 Teste de Desempenho

Após obter confirmação de que o sistema está funcionando corretamente em referência à documentação de requisitos do desenvolvedor, faz-se necessária a verificação dos requisitos de software e hardware, conforme utilização feita pelo cliente em seu ambiente de trabalho, desta maneira se o teste for realizado com sucesso, será produzido um sistema válido (PFLLEGER, 2004).

Outro teste de importante significância para o sistema é o teste de estresse, o qual é utilizado após total integração do sistema e baseia-se no teste de desempenho que deve assegurar que o sistema possa processar uma carga aceitável as necessidades do cliente, sendo que esta carga é aumentada até que o

sistema trave. Neste momento utiliza-se o teste de estresse que deve passar da carga máxima para provocar falhas, objetivando identificar defeitos que normalmente não ocorreriam e qual o comportamento do sistema em relação a este erro (SOMMERVILLE, 2003).

2.1.5 Teste de Aceitação

Esta atividade é realizada com a presença do cliente, na qual verifica se o sistema realiza o que foi definido pelo cliente na especificação dos requisitos, sendo realizado com sucesso verificar-se-á que o sistema foi aceito (PFLEEGER, 2004).

2.1.6 Teste de Instalação

Após o teste de aceitação o sistema deve ser instalado no ambiente de trabalho, devendo assim ser averiguado se o sistema ainda funciona devidamente, visando atender as necessidades do cliente (PFLEEGER, 2004).

Para exemplificação destes testes ao software, segue a figura 1 que representa a sequência lógica de realização deles:

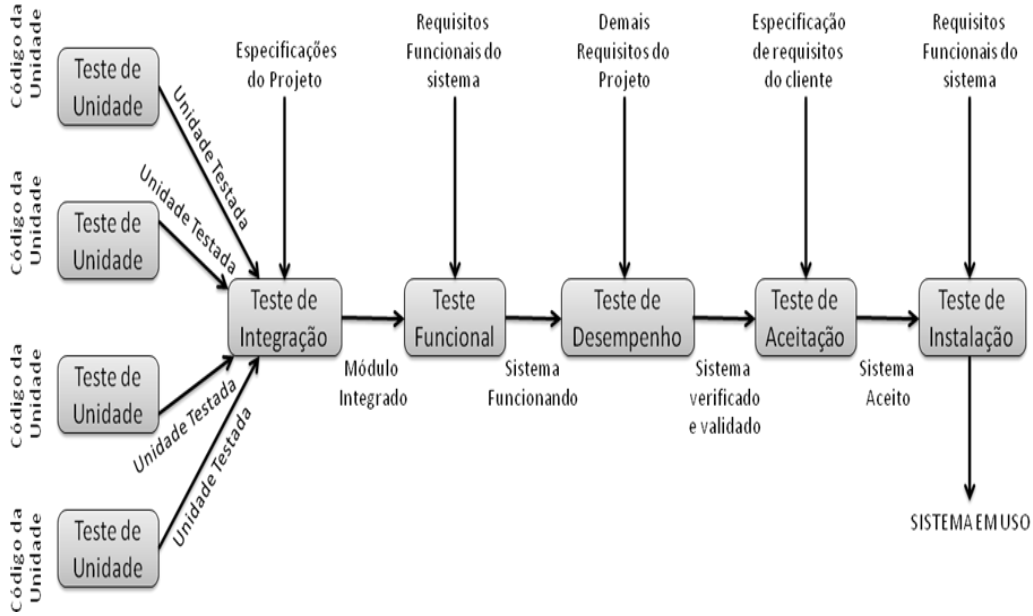

Figura 1: Etapas de Teste de Software (PFLEEGER, 2004).

Outro tipo de teste importante é o teste de regressão, porém ele não se caracteriza como um nível da atividade de teste, mas sim como uma estratégia adotada para a redução de “efeitos colaterais” ao software. Visto que esta estratégia pode ser realizada em qualquer nível de teste do software, sendo ela aplicada a cada nova versão do sistema, aplicam-se todos os testes que foram realizados na versão anterior, garantindo assim que soluções asseguradas na última versão estejam presentes nas futuras versões do software (NETO, 2007).

Esses diversos tipos de testes visão atingir em conjunto uma única meta principal: a qualidade. O próximo tópico explanará sobre a qualidade de software e porque é imprescindível a presença da qualidade em seu mais alto nível.

2.2 Qualidade de Software

A Engenharia de Software possui delimitação elaborada pelo SWEBOK (*Software Engineering Body Of Knowledge* ou Corpo de Conhecimento de Engenharia de Software), o qual visa especificar somente áreas de conhecimento referentes à Engenharia de Software, devido à grande abrangência de áreas e tecnologias existentes. Através dele tem-se que a qualidade é distinta em duas técnicas, sendo uma estática (seria a área de conhecimento qualidade, propriamente dita) e uma dinâmica (os testes), sendo assim ambas as técnicas visam obter uma maior qualidade (KOSCIANSKI; SOARES, 2006).

A maioria dos desenvolvedores de software acredita que pode começar a preocupar-se com a qualidade de seu produto somente após a codificação do software, o que está completamente errado, pois a qualidade deve ser gerenciada durante todo o processo de desenvolvimento (PRESSMAN, 2002).

Como explica Pressman (2002), se a equipe de desenvolvimento deseja entregar um software com qualidade, ela possui duas opções: fazer o software de maneira correta (assim preocupa-se com a garantia da qualidade desde o início do projeto), ou fazer novamente aquilo que de certa maneira não funcionou como o cliente desejava. O autor enfatiza ainda que na garantia da qualidade reduz-se o trabalho que deve ser refeito (diminuindo os custos) e o software estará no mercado em um menor prazo.

Segundo Rezende (2002) para que haja qualidade em um projeto é necessário: vontade de todos os integrantes da equipe, por se tratar de um processo

que deve ser executado durante todo o projeto; investimentos para a realização de tal processo; conhecimento de como garantir a qualidade; organização e planejamento para garantia.

Entretanto devido ao fato de os custos direcionados para a aplicação dos testes não serem considerados ideais ou satisfatórios e também pelo fato de o tempo disponível para a execução dos testes não ser o mais adequado ou mesmo que suficiente (pois poderia ocasionar atraso na entrega do produto), torna-se como admissíveis, na prática, a obtenção de uma qualidade e segurança para o software inferior a 100% (CALÇADO, 2007).

É fato que empresas que não garantem alta qualidade perdem clientes para as empresas concorrentes, assim sendo as empresas que disponibilizarem maior qualidade ao software obterão maior clientela. Desta maneira a qualidade tornou-se a melhor e mais eficiente “arma” para a conquista de novos clientes, que ao comprarem um produto com qualidade criam um “laço” de fidelidade com a empresa; pois na atualidade é mais importante um produto que disponibilize confiança ao cliente do que um produto mais barato ou que possa ser entregue em pouco tempo (WELCH, *apud* CALÇADO, 2007).

A qualidade requerida para um software está diretamente ligada com a garantia de que o produto não possui uma quantidade elevada de erros, assim pode-se possuir o mínimo aceitável de defeito, sendo que este não deve influenciar na execução do software. A quantidade de erros encontrados durantes os testes para a garantia da qualidade não deve ser levado em consideração, pois eles foram identificados pelos testes e possivelmente foram solucionados (BOAVENTURA, 2001, *apud* CÓCARO, 2005).

Para que um software seja considerado de qualidade, ele deve estar conforme os requisitos que foram levantados junto ao cliente; ser adaptável ao uso do cliente; estar adequado ao cliente e suas necessidades; estar sem defeito ou com defeitos que não impeçam a execução do software; que se enquadre no orçamento disponibilizado para desenvolvimento do projeto; que possua segurança nas informações geradas; que seja entregue dentro do prazo estipulado; assim visando satisfação do cliente (REZENDE, 2002).

Rezende (2002) reforça ainda que é relevante que o software de qualidade esteja adequado ao cliente e atenda suas necessidades, dispondo informações de

qualidade, sendo uma informação adequada, útil, confiável, clara, oportuna, relevante, segura, satisfatória para o cliente.

Conforme Calçado (2007) para garantir a qualidade do software, alguns critérios são necessários, tais como:

- Efetividade: garantia de que os objetivos propostos serão realizados com custos baixos;
- Pessoal: que seria um parâmetro entre a real qualidade do software desenvolvido e as expectativas que foram previamente estabelecidas pelo cliente;
- Custo: garantia de que os recursos destinados para realização do software não ultrapassaram o orçamento utilizado para confecção do produto;
- Manutenibilidade: garantia de que após a entrega ser feita pela equipe de desenvolvimento, todo e qualquer falha poderá e será solucionada pela equipe, dispondo ao cliente maior confiança no produto entregue;

De maneira geral a qualidade do software poderá ser definida em três pontos centrais que são de extrema importância e relevância não só para as empresa de software assim como para as empresas das diversas áreas do conhecimento, sendo eles: a satisfação do cliente com o produto adquirido (qualidade), a produtividade da equipe de desenvolvimento (entrega no prazo) e preocupações com os custos para a realização do projeto (orçamento) (NITA, 2007).

2.3 Problemas do Teste Convencional

O modelo convencional de desenvolvimento de software inicia-se com a fase de levantamento de requisitos perante o cliente, passando assim para as fases de design, codificação e testes, sendo que estes formam um ciclo iterativo de desenvolvimento de software, porém na atualidade não se pode considerar os requisitos levantados inicialmente como estáveis, visto que durante o desenvolvimento do software o cliente sempre tem a intenção de alterar os requisitos (SENGUPTA, 2004).

Esse fato ocorre com certa frequência devido à rápida evolução que atinge os negócios, visto que seus ciclos estão cada dia menores, assim ocorre de o desenvolvedor começar a codificação e os requisitos do cliente já mudarem devido ao negócio que eles estão associados (SENGUPTA, 2004).

Para Sengupta (2004), a realização destas mudanças provocará muito re-trabalho para a equipe de desenvolvimento que deverá alterar o código para que satisfaça a atual necessidade do cliente, agregando-se mais problemas ainda quando o desenvolvimento é feito de maneira distribuída, tanto fisicamente quanto temporalmente, uma vez que a alteração realizada em um módulo poderá provocar uma quantidade elevada de erros no momento da integração dos módulos.

Dessa maneira a realização de mudanças nos requisitos pode tornar-se muito caro ou inviável com o passar do tempo no desenrolar das etapas do desenvolvimento do software. A figura 2 apresenta o custo de mudanças ao longo do ciclo de desenvolvimento. (KOSCIANSKI; SOARES, 2007).

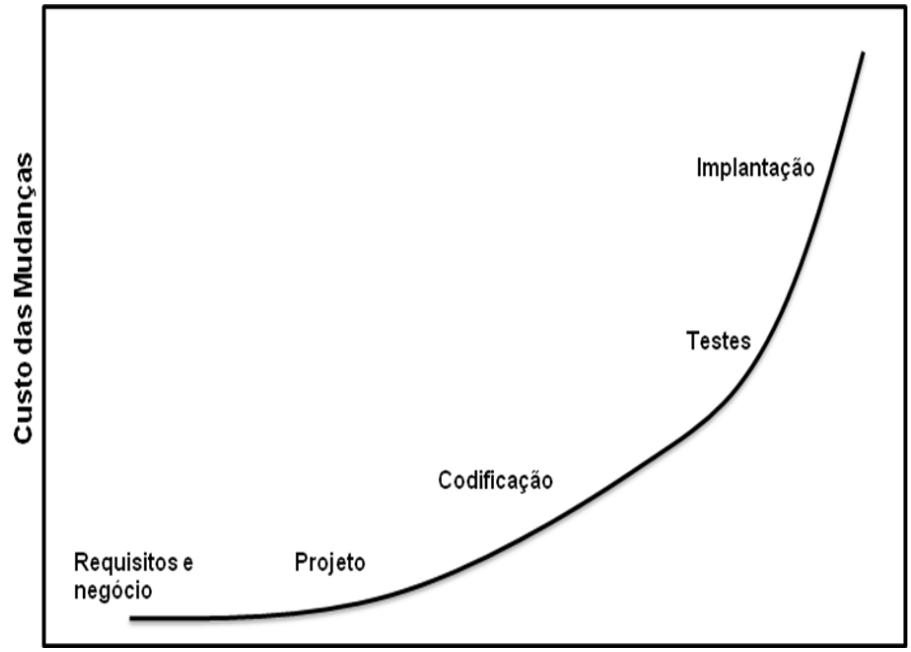

Figura 2: Custo de mudanças ao longo das etapas de desenvolvimento (KOSCIANSKI; SOARES, 2007).

Sommerville (2003) explica que a realização de várias etapas de testes é muito importante, pois a realização de um teste subsequente, como por exemplo, o teste de integração em relação ao teste de unidade, tem por objetivo encontrar erros referentes à integração dos módulos, mas também deverá identificar defeitos que não foram identificados na atividade de teste de unidade.

Fica explícito então que mesmo sendo realizados vários tipos de testes, não há garantia de que cada tipo de teste poderá identificar todos os defeitos ao qual foi destinado, assim pode-se gerar a identificação de um defeito que ocorre em qualquer módulo, somente no teste de instalação, o que vai necessitar de muito

mais tempo e gasto de recursos para sua solução, assim afetará as três metas de um projeto (qualidade, prazo e orçamento).

Pode-se definir que quanto mais tarde for identificado o defeito, mais tempo e recurso ele necessitará para ser resolvido, a figura 3 apresenta a relação entre o tempo e custo.

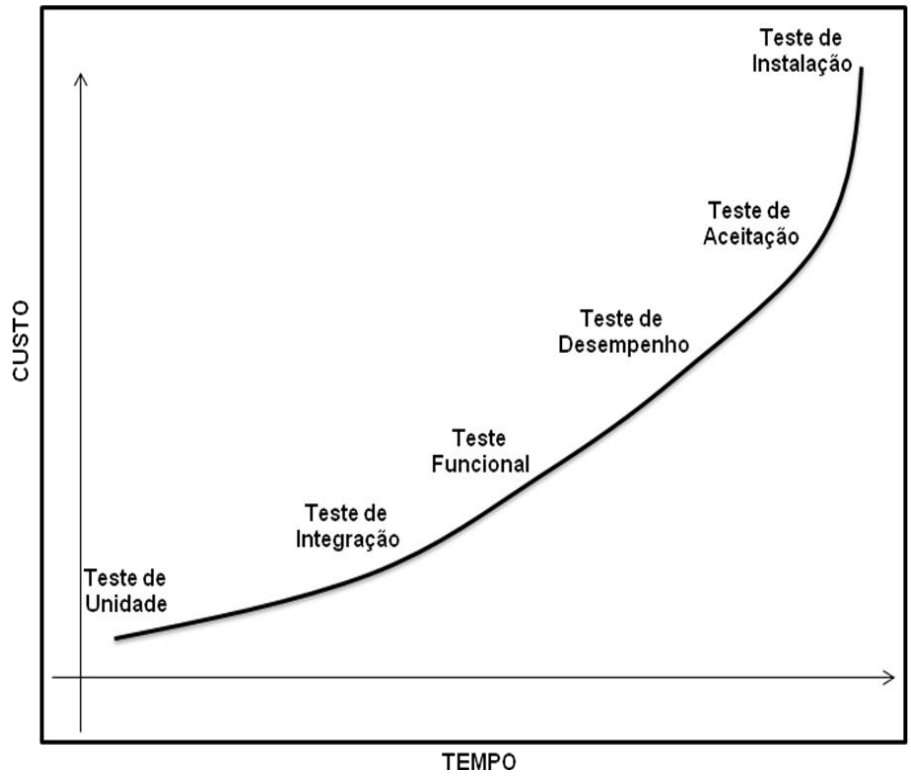

Figura 3: Resolução de Defeitos encontrados ao longo das atividades de teste.

Outra desvantagem da atividade convencional de teste é que ela visa somente à validação do código que foi elaborado, enquanto a prática de desenvolvimento orientado a teste, visa não somente à validação do código, como também serve de subsídio, assim como uma documentação, para os desenvolvedores que iram codificar o que foi requerido de funcionalidade do sistema pelo cliente (SENGUPTA, 2004).

Pode-se assim identificar no que consiste esta nova prática que objetiva a obtenção de melhores resultados em relação à atividade convencional de teste.

2.4 Test – Driven Development (TDD)

Para Sengupta (2004), uma solução para a ocorrência de mudança de requisitos é a prática do *test-driven development* (TDD) - desenvolvimento dirigido a teste. Para realização dessa prática devem-se criar casos de testes que serão elaborados antes da realização da codificação. O caso de teste se caracteriza por conter um conjunto de condições aplicadas a uma funcionalidade do software, visando definir quais seriam as saídas corretas de uma funcionalidade, assim que forem inseridas entradas corretas para execução de tal funcionalidade.

Assim no TDD primeiramente devem ser escritos casos de testes, antes mesmo da implementação de uma nova funcionalidade, desse modo, após eles serem elaborados, inicializa-se a codificação baseada nos casos de teste referentes à nova funcionalidade do sistema, que servirão de subsídio, como uma documentação para que o código possa ser implementado (SENGUPTA, 2004).

Conforme Daibert (2008), a prática do TDD utiliza as técnicas de testes para que o software possua, se possuir, a menor quantidade possível de erros, sendo que eles não influenciem na execução do sistema, para garantir o maior nível possível de qualidade.

Daibert (2008) reforça que todo o processo de desenvolvimento de software deverá basear-se nas técnicas de teste aplicáveis ao software, iniciando-se na técnica de se desenvolverem os testes antes da real implementação do código, chamada de *test first development* (TFD) – desenvolver primeiramente os testes. Assim o responsável por executar os testes deverá elaborar os casos de teste em referência a uma nova funcionalidade do sistema ou em relação a um módulo do sistema, antes deste ser codificado.

Os casos de teste disponibilizarão aos desenvolvedores uma compreensão mais detalhada e precisa em relação aos requisitos do software, visto que estes estarão mais explícitos e detalhados durante a elaboração dos casos de testes, sendo que as saídas corretas são geradas a partir das entradas corretas (SENGUPTA, 2004).

Na utilização da prática de TDD os casos de teste são otimizados, pois eles ocorrem de maneira incremental, acompanhando a etapa de implementação do código (DAIBERT, 2008).

Segundo Daibert (2008), a técnica de TFD utiliza como estratégia a programação por intenção que é caracterizada por induzir a programação de classes, métodos, variáveis, funcionalidades e até mesmo módulos que ainda não existem, mas deveram ser implementados futuramente para atender as necessidades de implementação que passaram a existir devido à necessidade de se executar uma nova funcionalidade do sistema, assim incentiva-se a criação de casos de teste.

Daibert (2008) define que a utilização desta estratégia de programação por intenção poderá e deverá disponibilizar a equipe de desenvolvimento quais serão os métodos a serem implementados para garantir eficiente execução da funcionalidade, define-se o que é essencial ser codificado; assim podem-se determinar quais os passos a serem seguidos para desenvolver uma funcionalidade que atenda as necessidades definidas pelo cliente e elaborados nos casos de teste.

A utilização do TDD garantirá não somente uma aplicação completa, como também uma quantidade considerável de testes de regressão efetivos (sendo que estes serão realizados na versão mais recente do software), ou seja, juntamente com o TDD durante a codificação, assim garante-se que não haverá nenhum defeito no software que já foi solucionado anteriormente, ou seja, o software não regredirá (SENGUPTA, 2004).

A especificação gerada através da utilização de casos de testes - demonstrará “o que deve ser implementado” - é mais rústica do que uma especificação completa, que é habitualmente realizada nos métodos convencionais de desenvolvimento, porém a especificação elaborada através dos casos de testes não requer qualquer outra habilidade por parte do desenvolvedor que deverá implementar só o que é tido como essencial para que a funcionalidade funcione e foi elaborado pelo caso de teste (SENGUPTA, 2004).

Relacionando-se à riqueza desta especificação pode-se levar em conta que esta especificação será enriquecida e ficará cada vez mais completa conforme serão inseridos novos casos de testes necessários. Visto também que o tempo disponibilizado para elaboração desta especificação é ínfimo em relação ao tempo que originalmente é disponibilizado para se levantar os requisitos no método convencional de desenvolvimento (SENGUPTA, 2004).

Devido à prática de TDD executar a elaboração dos casos de teste para servirem de subsídio durante a implementação do código do software, pode-se dizer

que os casos de teste realizados anteriormente a codificação servirão como uma documentação precisa perante as necessidades do software, conforme requisitado pelo cliente e em relação ao comportamento de cada módulo do projeto (SENGUPTA, 2004).

Como foi visto anteriormente na definição de Daibert (2008), para que seja utilizada efetivamente a prática do TDD é necessária a utilização da técnica de TFD, a qual pode ser exemplificada na figura 4:

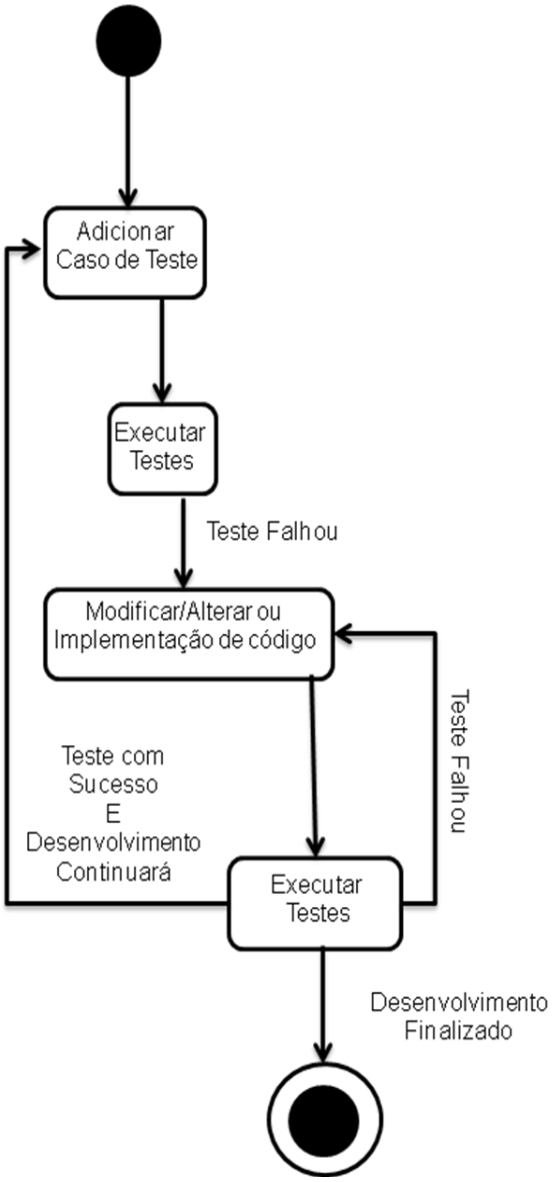

Figura 4: Passos para realizar o TFD (Daibert, 2008).

Assim pode-se, conforme Daibert (2008), detalhar o que ocorre em cada etapa da figura 4 como os passos a serem seguidos para efetiva execução do TFD, sendo que em cada passo deve ser realizada tal atividade:

1. Adicionar Caso de Teste: nesta atividade o teste ao software será escrito para que possa ser executado no passo seguinte;

2. Execução do Teste: na primeira vez em que for executado, certamente o teste será definido como falho, visto que não há nenhum código implementado a tal funcionalidade do software, assim justifica-se a existência do próximo passo;

3. Modificação, Alteração ou Implementação do Código: caso seja executado pela primeira vez, pode-se definir que nesta atividade dever-se-á implementar o código que supostamente irá satisfazer a necessidade da funcionalidade. Após a primeira execução, esta atividade deverá ser executada para modificar ou alterar o código que falhou após execução do teste, assim ocorrerá quando o teste demonstrar que há algum erro na funcionalidade, assim sendo alterado para que o teste possa ser executado com sucesso;

4. Execução do Teste: esta atividade possui o mesmo objetivo da atividade 2, porém ela é executada após a realização da atividade 3, a fim de garantir que as alterações executadas surtam efeito positivo na execução dos teste. Assim podendo atribuir: se o teste falhou e necessita retornar a atividade 3; se o teste obteve sucesso e o desenvolvimento irá continuar ou se o desenvolvimento foi finalizado.

Resumidamente, tem-se tal descrição de 10 segundos sobre TDD:

> Desenvolvimento orientado a testes significa que você escreve um teste automatizado, então escreve apenas código suficiente para fazê-lo passar, então você refatora o código primeiramente para melhorar a legibilidade e remover duplicação. Enxágüe e repita (KNIBERG, 2007).

A prática de TDD surgiu nas metodologias ágeis de desenvolvimento (que serão abordadas mais detalhadamente no próximo tópico), que segundo Koscianski e Soares (2007) são caracterizadas principalmente por serem aplicadas a projetos que trabalham com requisitos dinâmicos que possam ser alterados no decorrer da implementação do software, visto a rápida alternância de necessidade do cliente, essa que atualmente ocorrem com maior frequência.

Desta maneira Sengupta (2004) explica que em relação à constante mudança de requisitos que ocorre na utilização de TDD, deve-se atentar em relação aos casos de teste que:

- Caso um novo requisito seja adicionado, deve-se elaborar um caso de teste a ele e mapear as exigências deste, assim objetiva-se uma maior facilidade na realização da codificação;
- Caso um requisito seja excluído, deve-se também excluir o caso de teste referente ao requisito, pois o caso de teste estará obsoleto;
- Caso um requisito seja alterado, deve-se também alterar o caso de teste referente a ele, assim objetiva-se modificar as exigências referentes ao caso de teste, para que durante a codificação seja possível elaborar um software que garanta a satisfação do cliente;

Distingue-se assim consideravelmente do desenvolvimento tradicional, pelo fato de os casos de testes serem realizados antes da codificação, sendo que toda e qualquer mudança deve primeiramente ser aplicada aos casos de testes, por eles serem predecessores da fase de elaboração do código fonte, deve-se desta maneira atualizar inicialmente os casos de testes para verificação do impacto gerado pelas mudanças no sistema (SENGUPTA, 2004).

Desta maneira pode-se definir que a prática de TDD é muito difícil de ser efetuada, sendo que em muitos casos não importa o quanto você estudou a respeito do tema, para que um programador ou testador tenha prática em realizar TDD eficientemente é necessário que ele programe em conjunto com uma pessoa que já tem experiência e é bom na utilização de TDD, assim pode-se aprender como realizar tal prática (KNIBERG, 2007).

No entanto, a utilização da prática de TDD não minimiza a importância da realização de testes após a fase de desenvolvimento do software, visto que os testes realizados em fase inicial gerarão informações mais precisas para a equipe de desenvolvimento do código fonte, o que por consequencia diminuirá a ocorrência de defeitos depois do desenvolvimento (SENGUPTA, 2004).

2.4.1 Metodologias Ágeis

Para Koscianski e Soares (2007), uma metodologia de desenvolvimento de software é definida como um conjunto de atividades e processos que são executados para auxiliar a produção de um software.

As metodologias ágeis fazem frente perante às metodologias tradicionais de produção de software, visto que estas são orientadas à documentação para ser realizada a codificação de um software, como ocorre no modelo cascata, o que de certa maneira reprime e limita o desenvolvimento a ser realizado pelos desenvolvedores (SOARES, 2004).

Evidencia-se também que é “pesada” a utilização das metodologias tradicionais para várias organizações, principalmente as de pequeno e médio porte que não possuem orçamento viável ou não desejam a utilização de uma metodologia de produção com tantos processos. Assim algumas destas empresas optam por desenvolverem seus softwares sem a utilização de uma metodologia de desenvolvimento, o que provavelmente provocará efeitos desastrosos em relação à qualidade, orçamento e prazo de confecção do software, assim não atingirá as três metas de um projeto (SOARES, 2004).

A utilização de uma metodologia ágil enquadra-se em projetos de software que possuem curto prazo de entrega, com pequenas equipes de desenvolvimento e que objetivam um rápido desenvolvimento do software, visando à utilização de requisitos dinâmicos, os quais podem ser alterados no decorrer da implementação do software, sendo que o método tradicional de desenvolvimento requer um maior planejamento, no qual os requisitos são bem definidos e alguns futuros requisitos são esperados (SOARES, 2004).

Segundo Koscianski e Soares (2007) para que realmente se considere um desenvolvimento ágil, é necessário que haja uma mudança frequente nos requisitos, deve-se assim aceitar a ocorrência de mudanças, e não tentar prever o futuro, com análise de requisitos que poderão ser importantes.

Ao basear-se na definição de Soares (2004) outro diferencial presente nas metodologias ágeis está nelas possuírem enfoque nas pessoas e não nos processos de desenvolvimento de software, visto que há uma preocupação maior em disponibilizar mais tempo para a resolução de problemas de forma iterativa, ao invés de gastar este tempo com a documentação.

Mesmo com a evidente rapidez e agilidade observada na utilização das metodologias ágeis, há alguns processos que são tidos como fundamentais para elaboração de um software (SOMMERVILLE, 2003):

- Especificação do Software: atividade na qual serão definidos os requisitos do software, suas funcionalidade e restrições, é nessa atividade que a

equipe de desenvolvimento obtém um maior contato com o cliente, assim objetiva-se atender suas necessidades.

- Projeto e Implementação de Software: a partir dos requisitos estabelecidos com o cliente, a equipe de desenvolvimento pode começar a fase de codificação deste software, a basear-se nas funcionalidades que foram definidas pelo cliente, sendo que nesta fase são elaborados modelos através de diagramas e assim estes são implementadas através de alguma linguagem de programação.
- Validação do Software: nesta fase o software passará por uma conferência na qual será analisado se o software possui todas as funcionalidades que foram pré-estabelecidas pelo cliente, assim deve-se garantir que este pré-suposto seja válido e que realmente ocorra.
- Evolução do Software: atividade objetiva a constante evolução do software para que ele venha a atender as necessidades do cliente, visto que o software deve ser útil ao cliente e favorecerá sua maior satisfação.

Soares (2004) explica que o termo “Metodologias Ágeis” se tornou mais conhecido e popular em 2001 quando dezessete especialistas em processos de desenvolvimento de software, dentre eles *Scrum* e *Extreme Programming* (XP), estabeleceram quais seriam os princípios comuns entre ambos os processos. Formando assim o “Manifesto Ágil”, para o qual se obtêm tais conceitos chave [*Agile Manifesto* (2004) *apud* Soares]:

- Indivíduos e interações que se equiparam perante o método tradicional aos processos e ferramentas.
- Software executável que corresponderia à documentação no método tradicional.
- Colaboração do cliente correspondendo à negociação de contratos como prática comum no modelo de desenvolvimento tradicional.
- Respostas rápidas às mudanças que se contrapõem a seguir os planos como é realizado por equipes que possuem o modelo tradicional de desenvolvimento.

Além destes conceitos chave que foram explanados, tem-se que o manifesto ágil segue doze principio básicos, tal como (BECK *et al.* 2001):

- Prioridade maior na satisfação do cliente, com a entrega continua e adiantada em referencia ao prazo, de um software com valor;

- As mudanças nos requisitos são aceitas em qualquer etapa do desenvolvimento, até mesmo na etapa final, tudo para que o cliente possa tirar vantagem em relação à concorrência;
- Entrega de software que funcione, mesmo que com pequeno módulo, em pequenas escalas de tempo, como semanas ou no máximo um mês;
- Toda a equipe do projeto deve trabalhar junta do início ao fim do projeto, assim devendo haver comunicação entre a equipe de implementação e pessoas relacionadas ao negócio (contato com o cliente);
- Motivar sempre a equipe de desenvolvimento, dando a eles, um ambiente e suporte propícios para construção do software, sempre confiando que a equipe realizará seu trabalho;
- Utilização em grande escala da conversa pessoalmente, visto que a conversa “cara a cara” é o método mais eficiente e eficaz de se transmitir informações para, e dentro do time de desenvolvimento;
- Um software funcional é a medida primária para o progresso;
- O ambiente de desenvolvimento é sustentável, visto que desenvolvedores e usuários devem obter a capacidade de manter mesmo que indefinidamente passos constantes para evolução do projeto;
- Alto grau de prioridade e atenção deve ser disponibilizado a excelência técnica e a um bom design, assim aumentando a agilidade;
- Trabalhar com o principio da simplicidade, que prega a arte de maximizar a quantidade de trabalho que não precisou ser feita, assim fazendo somente o que é tido como essencial para o funcionamento;
- Busca por um time de desenvolvimento auto-organizável, que irá gerar a utilização da melhor arquitetura no desenvolver do projeto, requisitos bem implementados e design agradável ao cliente;
- Disponibilizar intervalos de tempo os quais servirão para o time de desenvolvimento refletir como pode ficar mais efetivo e eficaz, podendo assim se ajustar e otimizar seu comportamento de acordo com a melhor maneira para uma produção mais eficiente.

Soares (2004) ressalta como característica importante da metodologia ágil, o fato ser adaptável durante o processo de desenvolvimento do software, assim é perceptível que ela poderá se adaptar a novos fatores que ocorrerem durante o

processo de desenvolvimento do software, diferenciando-se assim das metodologias tradicionais que priorizam uma análise mais minuciosa e detalhada para averiguar previamente o que pode ocorrer e acontecer no decorrer da fase de desenvolvimento.

Esta característica de adaptação presente nas metodologias ágeis faz-se de extrema necessidade e vantagem na atual realidade das empresas de desenvolvimento de software, a qual requer uma maior possibilidade de mudança do software, pelo fato de as necessidades do cliente, na maioria das vezes, não serem muito claras nem mesmo para ele, o que poderá e possui uma grande parcela de chance de ser modificada pelo cliente no decorrer do processo de desenvolvimento do software, pois ele terá uma noção maior do software e perceberá algumas necessidades que não estavam tão claras no início do projeto (SOARES, 2004).

Segundo Soares (2004), algumas características presentes nas diversas metodologias ágeis diferem em termo de suas práticas propriamente ditas e das ênfases que são dadas em alguns processos, porém algumas características estão presentes e são compartilhadas em todas as metodologias ágeis, sendo comum a estas: um desenvolvimento iterativo e possivelmente com a integração de novos requisitos (incremental), uma maior ênfase na comunicação entre a equipe de desenvolvimento e a redução na elaboração de produtos como uma documentação extensa.

Conforme pesquisa realizada por Amber (2007) *apud* Banki e Tanaka (2008), pode-se notar a ampla adoção de uma metodologia ágil para o desenvolvimento de software:

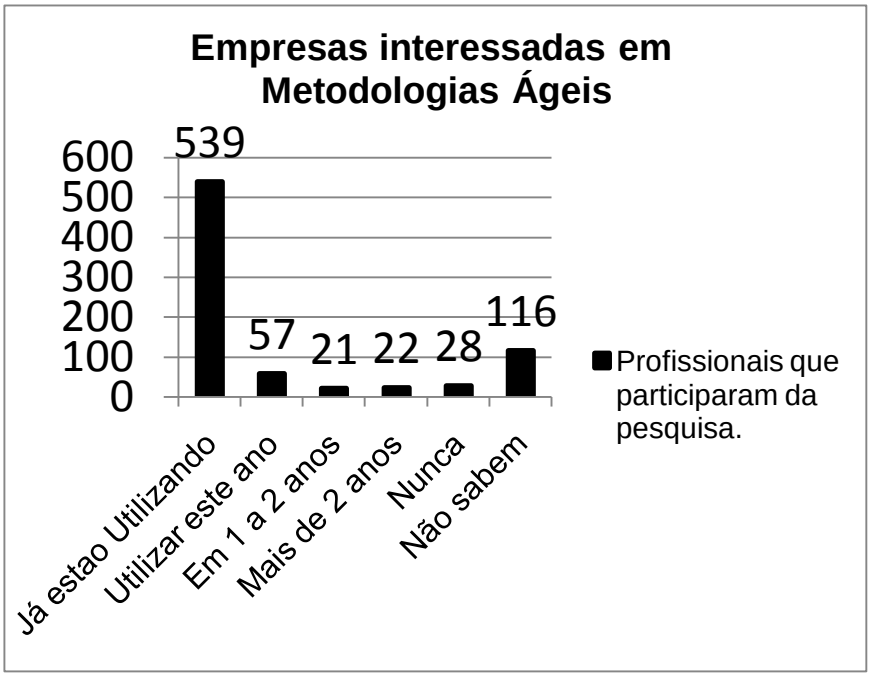

Figura 5: Prazo para adoção de uma metodologia ágil. [Amber (2007) *apud* Banki e Tanaka (2008)].

2.4.1.1 SCRUM

Segundo Fonseca e Campos (2008), *Scrum* enquadra-se melhor como um *framework* de processo ágil, que objetiva gerenciar o processo de desenvolvimento de software, através da utilização de práticas iterativas e incrementais.

Sendo que este desenvolvimento iterativo e incremental baseia-se em uma estratégia de planejamento das atividades a serem realizadas para conclusão do software, assim o software deve ser construído em partes que são realizadas uma em cada iteração. Desta maneira, ao fim de cada iteração, uma nova funcionalidade é incorporada ao software, sendo incrementado até o software estar completo (SANTOS, 2009).

O foco da metodologia *Scrum* está na tentativa de disponibilizar uma forma de trabalho flexível para a equipe de desenvolvimento de software produzir um sistema em um ambiente de trabalho, o qual está propício a ocorrência de frequentes mudanças (KOSCIANSKI; SOARES, 2007).

Desta maneira propõe-se uma forma de trabalho flexível que poderá adaptar-se a um ambiente dinâmico de desenvolvimento, pode-se assim conviver e tratar de questões como: mudanças nos requisitos do sistema; redução de período

do prazo de desenvolvimento; redução do orçamento disponibilizado para a confecção do software; alterações nas ferramentas e tecnologias que são utilizadas na elaboração, codificação e conclusão do software como um todo; trocas ou perdas de membros da equipe do projeto; dentre outros incidentes imprevisíveis que podem ocorrer durante o desenvolvimento do software (KOSCIANSKI; SOARES, 2007).

Segundo Koscianski e Soares (2007), esta metodologia baseia-se em princípios que são semelhantes aos encontrados em outras metodologias ágeis, tais como a *Extreme Programming* (XP)*,* sendo estes princípios: o desenvolvimento em equipes pequenas (de até dez pessoas), utilização de requisitos instáveis e muitas vezes desconhecidos, apresentando interações curtas que objetivam disponibilizar ao usuário uma visibilidade mais rápida e contínua do produto que está sendo construído.

No entanto, a *Scrum* é caracterizada por seus princípios próprios, dentre os quais se podem citar a utilização de *sprints.* Sendo que o *sprint* é uma iteração, na qual a equipe de desenvolvimento deve realizar a confecção de parte do software, assim objetiva-se produzir a entrega de valor para o cliente, visto também que cada iteração deve durar entre 2 e 4 semanas (SANTOS, 2009).

A *Scrum* possui somente três papéis, os quais são definidos por Santos (2009) com suas respectivas responsabilidades:

- *Product Owner* (PO)*:* membro é responsável por passar a equipe de desenvolvimento uma visão a respeito do produto a ser feito; assim deve-se elaborar uma lista com as funcionalidades necessárias para o software (*Product Backlog*) e a prioridade com que estas atividades devem ser feitas; deve-se também ser um representante do cliente dentro da equipe de desenvolvimento, sendo responsável por aceitar ou rejeitar a entrega de valor gerada pela equipe após uma iteração, antes que o produto seja entregue ao cliente;
- *Scrum Master:* é um líder perante a equipe de desenvolvimento, que deverá desta maneira, servir a equipe o que for necessário para realizar a produção do software, remover todos os impedimentos que atrapalham a evolução de produção do software; sendo responsável também por auxiliar o *product owner* na elaboração da lista de funcionalidades do software (*product backlog*) e garantir que as práticas do *scrum* estejam sendo executadas;
- *Team Scrum:* é a equipe de desenvolvimento (programadores, testadores, gerente banco de dados, gerente de interface, etc.) que deve possuir

entre 4 e 10 integrantes, sendo eles responsáveis por realizar a estimativa de tempo e esforço para realização das atividades, além de definir quais as tarefas devem ser feitas em cada atividade; sendo esta equipe responsável por produzir o software e garantir a qualidade total, isto porque ao fim do projeto, eles deveram apresentar o produto confeccionado ao cliente;

Segundo Santos (2009) na utilização da *Scrum*, algumas cerimônias são realizadas com o objetivo de manter o maior sincronismo e transparência possível para todos os membros da equipe:

- Reunião de Planejamento do *Sprint:* deve durar até 8 horas, com a participação dos três papéis do *scrum*, ela é a primeira reunião tem como objetivo planejar os *sprints.* Ela é dividida em duas partes, visto que na primeira parte o PO define a prioridade das atividades do *backlog* que são por ele selecionadas, assim define-se também a meta do *sprint;* na segunda parte a equipe define uma lista de tarefas as serem feitas em um *sprint* (*Sprint Backlog*);
- Reunião Diária: duração de 15 minutos, com a participação da equipe do *scrum master,* de preferência a reunião é realizada de pé, tem por objetivo que todos respondam três questões: o que eu fiz ontem; o que eu farei hoje; quais os impedimentos que eu encontrei;
- Revisão da *Sprint:* duração de 4 horas, com a participação dos três papéis e de convidados, se necessário, ela ocorre no fim de cada *sprint,* tem por objetivo averiguar tudo o que foi feito durante este período e efetuar a entrega de uma parte funcionando ao PO, assim pode-se incrementar o software;
- Retrospectiva da *Sprint:* duração de 3 horas, com a participação da equipe e do *scrum master*, ela é realizada após a revisão da *sprint,* objetiva-se averiguar o que deu certo neste e o que deu errado neste período, para que estes sejam solucionados antes de iniciar-se o próximo *sprint*;

Conforme Santos (2009) a utilização do *Scrum* gerará tais documentos ou artefatos:

- *Product Backlog:* documento que possuirá a lista que conterá todas as funcionalidades requeridas para o produto, o qual é gerado e gerenciado pelo *product owner*, que irá priorizar quais as funcionalidades que devem ser realizadas primeiramente. Neste artefato também se pode gerar as estórias do usuário que se caracterizam por ser uma descrição detalhada de cada *product backlog,* sendo que

elas auxiliam no entendimento do desenvolvedor e na realização da estimativa de tempo e esforço através de métodos como *planning poker;*

- *Sprint Backlog:* este documento é composto por uma lista de tarefas que serão realizadas durante um *sprint,* assim objetiva-se que elas sejam realizadas a fim de atingir a meta de cada *sprint;*
- *Burndown:* este documento possibilita maior precisão para se gerenciar o desenvolvimento do software, visto que ele representa graficamente as tarefas realizadas em relação do tempo, assim pode-se medir se a equipe está com um nível de evolução maior do que se esperava, ou se a equipe está atrasada em relação à meta traçada para cada *sprint*, sendo que o gráfico é atualizado diariamente, assim facilita-se a tomada de decisão sobre como se poderá garantir que a meta do *sprint* seja atingida;

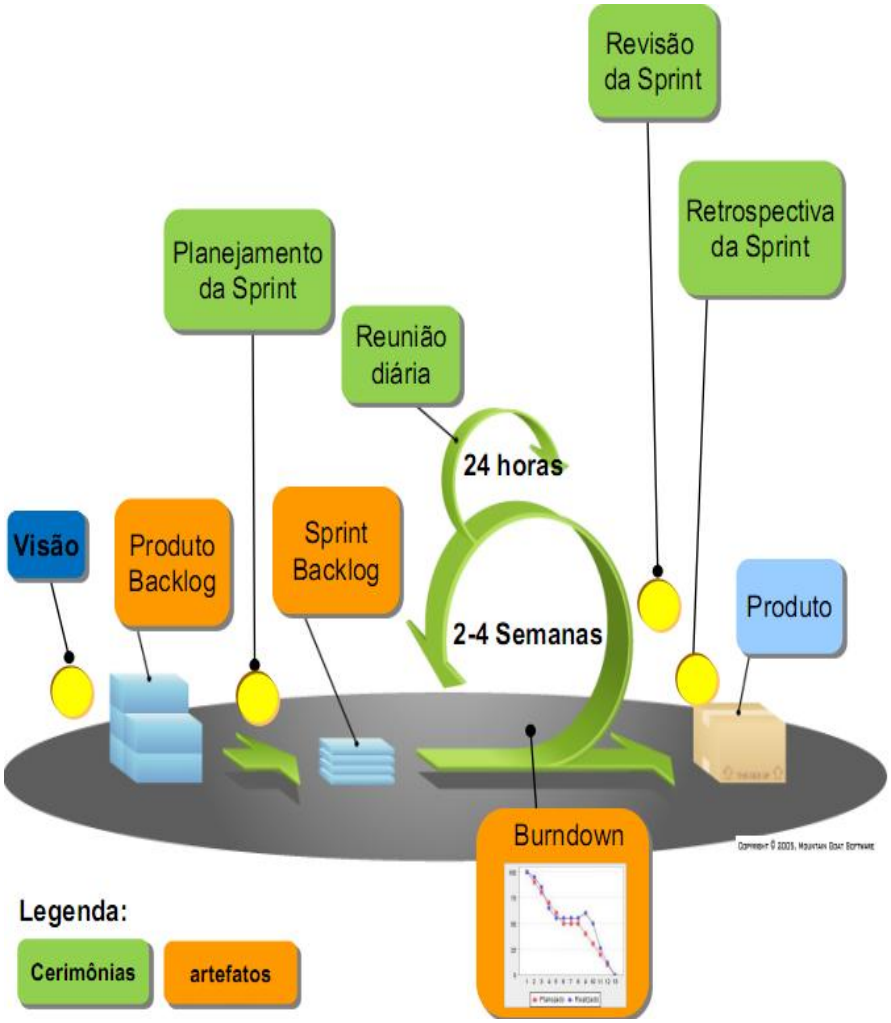

Figura 6: Estrutura geral do Scrum (SANTOS, 2009).

Segundo Santos (2009), para exemplificar o grande nível de adoção da utilização da *scrum*, tem-se que tais empresas trabalham com esta metodologia: Google, Microsoft, Yahoo, Siemens, Borland, Philips, Intel, BMC Software, Nokia, Nielsen, Canon, HP Invent, BBC, Amazon. Tendo em âmbito nacional: Globo.com, Revista Abril, UOL, PowerLogic, LocaWeb, Ci&T, dentre outras.

3 DESENVOLVIMENTO

Para o desenvolvimento do trabalho foi necessária a execução de algumas atividades precedentes a utilização da técnica de TDD e análise de seus benefícios, sendo este o objetivo do trabalho.

As atividades realizadas antes da utilização do TDD foram: primeira entrevista com o cliente e usuário final para entendimento e levantamento dos requisitos, elaboração da modelagem de negócio para melhor entendimento da equipe, segunda entrevista com o cliente e usuário final para tirar dúvidas ainda existes e melhorar o entendimento da equipe de desenvolvimento.

Após serem definidos os requisitos do software e elaborados os modelos de negócio pelo analista de interface, toda a equipe teve uma melhor visão do produto a ser elaborado. Desde então, o analista de testes teve de definir o modelo de casos de teste a ser utilizado na aplicação do TDD e iniciar a elaboração deles.

Desta maneira, o analista de testes realizou uma pesquisar com o objetivo de definir qual seria o modelo de caso de teste que melhor se adequava a equipe de desenvolvimento. Para isto, foram definidos dois modelos de casos de teste, no qual um seria utilizado para auxiliar a codificação do software e o outro seria utilizado para elaboração de casos de teste que iriam refatorar o código até que cada funcionalidade fosse concluída.

Sendo estes dois modelos de casos de testes definidos teve-se início a utilização da prática de TDD para o desenvolvimento do software. Para isso, serão apresentadas na sequência todas as atividades realizadas de maneira mais detalhada e precisa.

3.1 Definição dos Requisitos

Para realização deste trabalho, foi proposta a utilização da prática de *Test-Driven Development (TDD)* – desenvolvimento orientado a teste – como método para a implementação de um software que será desenvolvido para Adecot (Associação de Desenvolvimento Comunitário das Três Águas), assim disponibilizando ao responsável designado pela associação uma maior facilidade e

efetividade na realização das cotações que atenderam as aproximadamente 100 famílias associadas.

Para efetuar a elaboração deste software, a equipe de desenvolvimento necessitou obter uma maior compreensão de como o funcionário da associação realizaria as cotações dos produtos de maneira manual, sendo assim possível implementar um programa que pudesse efetuar suas ações de modo mais ágil, eficiente e confiável, otimizando a realização das cotações para associação.

Sendo assim, a primeira atividade a ser executada para o desenvolvimento do software foi à realização de uma entrevista diretamente com o funcionário da associação responsável por efetuar as cotações, com o objetivo de efetuar a definição dos requisitos necessários para se realizar a implementação do produto (SOMMERVILLE, 2003).

Esta entrevista foi realizada no dia 13 de agosto, contando com a presença quase que total da equipe de desenvolvimento, composta pelos dois gerentes de projeto, sendo um o *product owner* e o outro o *scrum master* (visto que a equipe trabalha com a metodologia ágil Scrum), um analista de interface, um programador e um analista de testes, que formam o *team scrum* com mais um programador e um estagiário. Completando os componentes da entrevista, estava o funcionário da associação, o qual seria entrevistado pela equipe de desenvolvimento.

A entrevista começou com um relato por parte do funcionário que demonstrou de maneira genérica, quais eram os passos executados por ele para que as cotações pudessem ser realizadas. Após esta fase inicial, os membros da equipe de desenvolvimento passaram a tirar algumas dúvidas que haviam ficado com o objetivo de obter uma maior compreensão em relação ao processo como um todo.

Após a equipe sanar suas dúvidas com o funcionário da associação e retornar ao ambiente de desenvolvimento foi realizada a modelagem de negócio pelo analista de interface exemplificando as atividades que eram executadas pelo funcionário, com o objetivo de definir quais seriam as funcionalidades a serem implementadas para o software, de maneira a atender todas as atividades que eram executadas pelo responsável da associação, e assim facilitar de modo considerável a realização delas.

Efetuada a modelagem das atividades realizadas pelo funcionário, surgiram novas dúvidas em relação a algumas propriedades e prioridades que poderiam ser

consideradas pelo funcionário, tais como: quais os dados que ele necessitava para realizar um pedido de um associado; se todos os produtos que formavam o pedido de um associado eram requisitados de uma única empresa ou de várias; se a cotação levava somente em conta os preços ou havia a distinção entre as empresas.

Desta maneira foi necessária a realização de uma segunda entrevista com o funcionário da associação, sedo esta realizada no dia 3 de setembro, com o objetivo de obter algumas informações mais detalhadas do processo executado pelo funcionário e informações mais especificas em relação aos dados que ele necessitaria para utilizar o software de maneira mais efetiva, ágil, confiável e com qualidade, assim dispondo a ele um serviço automatizado que se assemelharia com o serviço antes realizado manualmente.

Após ser efetuada esta segunda entrevista com o cliente, foi possível finalizar a etapa de levantamento de requisitos e aprofundamento em relação à compreensão da equipe sobre as especificidades do software.

3.2 Casos de Teste

Com o fim da etapa de entrevista e com a elaboração de uma modelagem de negócio produzida pelo analista de interface que contém todas as atividades realizadas no processo de cotação, teve-se início a etapa de elaboração dos casos de teste que devem servir de subsídio aos programadores, assim iniciando a utilização da prática de TDD (SENGUPTA, 2004).

A fase de elaboração dos casos de teste é uma etapa fundamental para a realização do desenvolvimento deste trabalho, porém a elaboração deles é de complicada definição e utilização, visto a pouca quantidade de materiais que auxiliam na confecção dos casos de teste disponibilizando a efetiva aplicação do TDD (KNIBERG, 2007).

Desta maneira o analista de testes da equipe de desenvolvimento analisou alguns exemplos de casos de teste e definiu um modelo que melhor se enquadra na equipe para que ela estivesse apta a desenvolver o software utilizando a prática proposta.

Para isto, o analista de teste optou por utilizar dois modelos de casos de teste diferentes para poder atingir os princípios da prática de desenvolvimento

orientado a teste. O primeiro modelo definido servirá para subsidiar a codificação do software a ser implementada, já o segundo modelo definido objetiva gerar informações e artefatos que sirvam para refatoração do código até que este esteja completamente implementado (SENGUPTA, 2004).

3.2.1 Caso de Teste para Implementação

O modelo que é inicialmente elaborado possui a finalidade de auxiliar na codificação das funcionalidades, informando ao programador o que ele deve implementar (SENGUPTA, 2004). Porém a primeira versão que será gerada por este modelo deverá conter somente informações sobre os campos requeridos para execução da funcionalidade e os botões que devem existir, ou seja, este primeiro caso de teste serve para mostrar ao desenvolvedor somente qual será a estrutura de tela da funcionalidade a ser desenvolvida.

Este modelo de caso de teste é composto basicamente por duas partes, baseando-se no modelo de casos de teste utilizados pela CELEPAR (MAYER; CHIARELLO, 2009). Primeiramente é formado um cabeçalho composto por informações de nomenclatura do caso de teste, que servirá para contextualizar o caso gerado e demonstrar algumas informações genéricas.

O cabeçalho é composto pelos campos: Caso de Teste (nome específico para sua identificação), Caso de Uso (demonstra qual o caso de uso do programa que possui relacionamento com o caso de teste), Responsável (analista de teste responsável pelo caso de teste e programador que deverá implementá-lo), Data (de elaboração do caso de teste), Tipo de Teste (com o objetivo de informar o que o teste deve verificar), Abordagem (execução de teste manual ou por meio de ferramenta), pré- condições, pós-condições e informações adicionais (MAYER; CHIARELLO, 2009).

Segundo Mayer e Chiarello (2009), o campo destinado a pré-condição deve informar quais os passos e etapas que devem ser executados pelo usuário até chegar à funcionalidade para a qual o caso de teste está sendo criado, assim preenchendo todos os pré-requisitos a serem realizados para que a funcionalidade testada possa ser executada corretamente (como a execução de um cadastro anterior a uma consulta).

Como indicado pelos autores, o item do cabeçalho referente à pós-condição deve indicar qual é a condição exigida após a correta execução da funcionalidade que esta sendo testada, ou seja, demonstrar qual é o resultado esperado descrevendo o que deve ser gerado pelo sistema após a funcionalidade ser executado corretamente (como o registro de um cadastro no banco de dados, a correta associação entre um pedido feito e um pagamento efetuado).

Por fim, o último item do cabeçalho a ser utilizado ou preenchido de modo opcional faz referência a inserção de alguma informação adicional ou comentário que o analista de teste responsável deseje registrar em relação ao caso de teste gerado que não foi possível inserir em outro campo (MAYER; CHIARELLO, 2009).

CABEÇALHO	
Caso de Teste	Controle de Orçamento
Caso de Uso Relacionado	Controlar Orçamento
Responsável	Analista de teste – Saulo Programador – André
Data	27/09/2010
Tipo de Teste	Teste de Validação
Abordagem	Realização de Teste Manualmente
Pré-condições	Usuário deve estar *Logado* no sistema e acessar a opção de orçamento. Anteriormente a esta execução é necessário que o usuário tenha cadastrado o fornecedor associado aos produtos orçados e ter cadastrado também os produtos relacionados para o orçamento.
Pós-condições	A cotação do preço de todos os produtos relacionados a um fornecedor deve ser registrada na base de dados, contendo uma relação entre o fornecedor (Domene), o ano de cotação (2010), um período do ano de cotação (2º semestre), o produto a ser orçado (Randap) e os valores do produto para dois meses (8,60) ou safra (9,90), estando estes dados disponíveis para consulta e alteração dos dados.
Informações Adicionais	Atentar para a necessidade de serem executados os cadastros do fornecedor e dos produtos antes da execução do orçamento.

Tabela 1 - Cabeçalho do Modelo de Caso de Teste de Auxílio para Implementação.

A segunda parte a ser elaborada é o corpo do caso de teste, que será composto por informações mais especificas em relação a cada caso de teste criado, que auxiliarão na implementação das funcionalidades do software. Segundo Mayer e Chiarello (2009) esta parte complementar do caso de teste possui o objetivo de descrever todas as ações que serão realizadas pelo usuário e as consequentes ações a serem executadas pelo sistema.

Desta maneira, esta parte complementar do caso de teste representará seu corpo, com os seguintes campos a serem preenchidos pelo analista de teste: o

cenário (formado pelo ator e pela descrição) e o resultado esperado (que deve informar qual o resultado que se espera obter, para que a execução da funcionalidade seja considerada correta) (MAYER, CHIARELLO, 2009).

O cenário a ser preenchido deve fazer referência ao caso de uso que possui relação com o caso de teste que será elaborado, sendo composto pelo ator (usuário final que utilizará o software) e pela descrição (que deverá informar quais serão as ações realizadas pelo usuário e as reações do sistema). Segundo Mayer e Chiarello (2009) a descrição deve abranger todos os eventos do caso de uso e utilizar dados reais para execução do software.

CORPO		
Cenário:		
	Ator	Responsável da associação
	Descrição	**1.** Ator acessa a página com o objetivo de cadastrar e controlar o orçamento de alguns produtos; **2.** Ator inicia o cadastramento do orçamento selecionando um fornecedor dos produtos cotados, preenchendo o ano do orçamento e selecionando o período do ano em que o orçamento foi realizado; **3.** Ator seleciona o produto que foi orçado com o fornecedor informado, inserindo os valores levantados para pagamento no período de dois meses e o valor pago para a safra (seis meses); **4.** Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir o novo produto orçado em uma tabela que será registrada na base de dados; **5.** Ator deverá confirmar o cadastro clicando no botão de SALVAR; **6.** O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Orçamento Cadastrado com Sucesso).
Resultado Esperado	Orçamento registrado no banco de dados relacionando o fornecedor com os produtos orçados e seus respectivos valores para a data de pagamento, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.	

Tabela 2 - Corpo do Modelo de Caso de Teste para Implementação.

Sendo elaborado este modelo de caso de teste que servirá para auxiliar os desenvolvedores na implementação do código do software, tem se início o desenvolver do programa. Após o programador utilizar este primeiro modelo, as telas iniciais do programa já estarão prontas e disponíveis para testes, assim iniciando a elaboração do segundo modelo de caso de teste utilizado, que objetivará gerar informações mais precisas aos programadores para que eles possam refatorar o código e enriquecer o software (SENGUPTA, 2004).

3.2.2 Caso de Teste para Refatoração

Com a elaboração do caso de testes para implementação, faz-se necessária a implementação do segundo modelo de caso de teste que terá como objetivo validar todas as ações realizadas dentro da funcionalidade testada, assim disponibilizando informações específicas que possibilitaram aos desenvolvedores refatorarem o código até que ele atenda a todas as validações necessárias (SENGUPTA, 2004).

O modelo de caso de teste validará quais seriam as saídas corretas de uma funcionalidade sendo inseridas entradas pertinentes para execução correta da funcionalidade, ou seja, para todo o software serão elaborados casos de testes que auxiliariam o analista de testes no planejamento da execução dos testes, e através destes serem geradas informações que implicaram na refatoração do código para que o programa atenda a necessidades analisadas, assim capacitando ao sistema gerar os resultados esperados que fossem pré-estabelecidos (SENGUPTA, 2004).

Para elaboração destes casos de teste foi necessário estabelecer alguns requisitos validadores que possibilitariam a refatoração do programa. Desta maneira, o analista de testes definiu treze (13) validadores que possibilitaram a criação dos casos de testes e consequentemente auxiliaram na refatoração do software.

VALIDADORES
1. Validação da seqüência na alternância com a tecla TAB
2. Validação quanto à inserção correta dos dados em relação a tipo de dado que os campos esperam
3. Validação da mensagem de erro de preenchimento
4. Validação da inexistência do novo cadastro, para que não haja duplicidade de dados
5. Validação do número limite de caracteres em todos os campos
6. Validação da obrigatoriedade de preenchimento para os campos necessário
7. Validação da existência de TITLE de auxilio em todos os campos
8. Validação da não inserção da quantidade igual a zero
9. Validação da veracidade do CPF
10. Validação do relacionamento no caso de duas datas
11. Validar as datas inexistentes
12. Validação dos pop-ups de confirmação das operações
13. Validar a quantidade de dados que a tabela suporta

Tabela 3 - Validadores utilizados para geração dos casos de teste para refatoração.

Estes validadores representam somente os requisitos que seriam inicialmente testados para que o código pudesse ser incrementado e enriquecido, porém outros validadores poderiam ser considerados, conforme as necessidades específicas de cada tela da aplicação, assim estes validadores representam somente os tópicos genéricos que geralmente eram validados pelo analista de testes.

Para exemplificar os casos de teste gerados a partir da validação destes requisitos, serão apresentados abaixo os casos de testes elaborados para funcionalidade de Controle de Orçamento.

CASOS DE TESTE PARA VALIDAÇÃO E REFATORAÇÃO	
Teste: Sequência de alternância de campos utilizando a tecla TAB. **Exemplo: <objeto selecionado; TAB; resultado esperado>**	Caso 1 <NOVO; TAB; CONSULTAR> Caso 2 <SAIR; TAB; Fornecedor> Caso 3 <Semestre; TAB; OK> Caso 4 <OK; TAB; Produto> Caso 5 <Safra; TAB; ADICIONAR> Caso 6 <Último item da tabela; TAB; NOVO>
Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas). **Exemplo: <campo; dado de entrada; resultado esperado>**	Caso 1 <Ano; Atual; Mensagem de erro> Caso 2 <2 Meses; 10 reais; Mensagem de erro> Caso 3 <Safra; 12 reais; Mensagem de erro>
Teste: Validação das mensagens para erro de preenchimento dos campos. **Exemplo: <campo; mensagem de erro>**	Caso 1 <Ano; Insira somente números. Ex. 2010> Caso 2 <2 Meses; Insira somente números. Ex: 10 00> Caso 3 <Safra; Insira somente números.
Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados. **Exemplo: <campo; mensagem de duplicidade>**	Caso 1 <Fornecedor, Ano e Semestre; Orçamento já cadastrado>
Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados. **Exemplo: <quantidade (nome do campo); resultado esperado>**	Caso 1 <4 (Ano); Correto> Caso 2 <5 (Ano); Erro> Caso 3 <3 (Ano); erro> Caso 4 <7 (2 Meses); Correto> Caso 5 <8 (2 Meses); Erro> Caso 6 < 7 (Safra); Correto> Caso 7 <8 (Safra); Erro>
Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos. **Exemplo: <campo sem nada; resultado esperado>**	Caso 1 <Fornecedor; Erro> Caso 2 <Ano; Erro> Caso 3 <Semestre; Erro> Caso 4 <Mercadoria; Erro> Caso 5 <2 Meses; Erro> Caso 6 <Safra; Erro>
Teste: Validação da existência de TITLES de auxílio em todos os campos. **Exemplo: <campo; mensagem de auxílio>**	Caso 1 <Fornecedor; Selecione o fornecedor dos produtos orçados. Ex: Domene> Caso 2 <Ano; Insira um ano valido. Ex: 2010> Caso 3 <Semestre; Selecione o semestre em que o orçamento foi realizado. Ex 2º Semestre>

	Caso 4 < Mercadoria; Selecione a mercadoria para inserção dos valores. Ex: Herbicida> Caso 5 <2 Meses; Insira qual o valor da mercadoria para ser paga em 2 meses. Ex: 10 00> Caso 6 <Safra; Insira o valor da mercadoria para ser paga na safra. Ex: 12 00>
Teste: Validar a não inserção de valor igual a zero. **Exemplo: <campo; Mensagem de erro>**	Caso 1 <Ano; Insira um ano diferente de zero> Caso 2 <2 Meses; Insira um valor diferente de zero> Caso 3 <Safra; Insira um valor diferente de zero>
Teste: Validar o ano do orçamento. **Exemplo: <dado de entrada; resultado esperado>**	Caso 1 <Ano que ainda não chegou; Mensagem de erro: Ano Inexistente> Caso 2 <Ano incompleto; Mensagem de erro: Ano Inexistente> Caso 3 <Ano correto; Correto>
Teste: Validação dos pop-ups de confirmação para as operações requeridas. **Exemplo: <botão; estado; mensagem>**	Caso 1 <SALVAR; correto; Orçamento Cadastrado com Sucesso> Caso 2 <OK; errado; Orçamento Já Cadastrado> Caso 3 <SAIR; correto; Orçamento não salvo. Deseja sair desta página?> Caso 4 <SAIR; errado; Orçamento não salvo. Deseja sair desta página?>
Teste: Validar a quantidade de dados que a tabela suporta. **Exemplo: <quantidade; resultado esperado>**	Caso 1 <5; primeira tabela> Caso 2 <6; segunda tabela> Caso 3 <11; terceira tabela>

Tabela 4 - Casos de Teste que auxiliam a refatoração do código.

Após serem elaborados estes casos de testes, o analista de teste passa a ter informações mais específicas em relação à funcionalidade, ou seja, a utilização dos validadores gera requisitos que possibilitam a refatoração do software (SENGUPTA, 2004). Estes validadores utilizados deveram corresponder às reações tomadas pelo sistema em relação às ações do usuário do programa, que foram elaboradas na descrição do modelo de casos de testes que subsidiam a implementação do software.

Desta maneira, o analista de testes deve fazer uma integração entre os dois modelos de casos de testes, utilizando as informações geradas pelos casos de teste que possibilitam a refatoração para incrementar os casos de teste que subsidia a codificação do programa.

3.2.3 Integração dos Modelos de Caso de Teste

Com a elaboração do primeiro modelo de caso de teste (que possibilitou a criação da tela referente às funcionalidades), foi possível executar o segundo modelo de casos de teste (que objetiva elaborar casos de testes que gerem informações necessárias para a refatoração do código) e com a execução do segundo modelo é possível incrementar o primeiro modelo.

Assim, devem ser integrados os dois modelos, servindo o segundo modelo para incrementar o primeiro modelo. O incrementar do modelo que serve de apoio aos desenvolvedores de software deverá ser feito aos poucos, para que, caso haja alguma modificação necessária a funcionalidade, esta não venha a causar uma grande regressão ao desenvolver do programa, ou seja, o analista de teste deverá incrementar as novas informações de maneira cadenciada (SENGUPTA, 2004).

Em relação à funcionalidade de Controle do Orçamento (anteriormente apresentada), serão inseridas informações que possibilitaram a refatoração pouco a pouco. Para exemplificar, segue abaixo o modelo final, contendo todas as informações que foram geradas na utilização de todos os validadores. Deve-se observar também que somente serão realizadas alterações no corpo do modelo que subsidia a implementação.

CORPO		
Cenário:		
	Ator	Responsável da associação
	Descrição	**1.** Ator acessa a página com o objetivo de cadastrar e controlar o orçamento de alguns produtos; **2.** Ator inicia o cadastramento do orçamento selecionando um fornecedor dos produtos cotados, preenchendo o ano do orçamento e selecionando o período do ano em que o orçamento foi realizado; **3.** Ator seleciona o produto que foi orçado com o fornecedor informando, inserindo os valores levantados para pagamento no período de dois meses e o valor pago para a safra (seis meses); **4.** Sistema deve garantir a correta alternância dos campos com a utilização da tecla TAB; **5.** Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data); **6.** Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos; **7.** Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados; **8.** Sistema deve validar o número limite de caracteres em todos os campos do software*;* **9.** Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos; **10.** Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do software; **11.** Sistema deve validar a não inserção dos valores igual a zero para os campos Ano, 2 meses e Safra; **12.** Sistema deve validar a inserção de um ano valido para o correto cadastramento; **13.** Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias; **14.** Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir o novo produto orçado em uma tabela que será registrada na base de dados; **15.** Sistema deve validar a correta inserção dos dados na tabela, para que não haja perda de qualquer dado; **16.** Ator deverá confirmar o cadastro clicando no botão de SALVAR; **17.** O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Orçamento Cadastrado com Sucesso).
Resultado Esperado	Orçamento registrado no banco de dados relacionando o fornecedor com os produtos orçados e seus respectivos valores para a data de pagamento, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.	

Tabela 5 - Corpo do Caso de Teste para Implemetação com as instruções para refatoração do código.

O desenvolvimento deste trabalho fica então baseado na execução das atividades de: elaboração dos casos de teste que servem de especificação para a criação das telas; criação dos casos de testes que são baseados em validadores

para geração de informações que possibilitem a refatoração do software; e a integração entre os dois modelos de casos de teste para que o desenvolvedor possa incrementar o programa até que este esteja completamente desenvolvido. Sendo assim, possível utilização a técnica de desenvolvimento orientado a testes.

4 RESULTADOS

Deve-se determinar que até o término do desenvolver deste trabalho, o software para o qual a técnica de desenvolvimento orientado a teste (TDD) foi aplicada, não foi concluído e continua a ser implementado pela equipe de desenvolvimento.

Para utilização de tal técnica, foram elaborados modelos de casos de teste para: criação das telas do programa; seguindo da validação das funcionalidades e conseqüente refatoração do software.

As tabelas abaixo apresentam a elaboração e implementação dos modelos de casos de teste, estando elas separadas pela finalidade de desenvolvimento de cada modelo, apresentando quais os casos de teste que devem ser elaborados, quais já foram elaborados pelo analista de testes e quais já foram implementados pelos desenvolvedores.

CASOS DE TESTE – Criação das Telas		
CASOS A SEREM ELABORADOS	**CASOS ELABORADOS**	**CASOS IMPLEMENTADOS**
Cadastro de Produtos	**X**	**X**
Cadastro de Produtores	**X**	**X**
Cadastro de Fornecedores	**X**	**X**
Cadastro de Usuários	**X**	**X**
Controle de Orçamento	**X**	**X**
Realização de Pedidos	**X**	**X**
Financeiro		
Relatórios		

Tabela 6 - Demonstração dos Casos de Teste para Elaboração e Implementação das Telas que foram desenvolvidos até o momento.

CASOS DE TESTE – Refatoração do Código		
CASOS A SEREM ELABORADOS	**CASOS ELABORADOS**	**CASOS IMPLEMENTADOS**
Cadastro de Produtos (1 - 3)	**X**	**X**
Cadastro de Produtos (4 - 6)	**X**	**X**
Cadastro de Produtores (1 -4)	**X**	**X**
Cadastro de Produtores (5 -7)	**X**	
Cadastro de Produtores (8 -10)	**X**	
Cadastro de Fornecedores (1 -3)	**X**	**X**
Cadastro de Fornecedores (4 -6)	**X**	**X**
Cadastro de Fornecedores (7 -9)	**X**	**X**
Cadastro de Usuários (1 -3)	**X**	**X**
Cadastro de Usuários (4 -6)	**X**	
Cadastro de Usuários (7 -9)	**X**	
Realização de Pedidos (1 - 4)	**X**	**X**
Realização de Pedidos (5 - 8)	**X**	
Realização de Pedidos (9 - 11)	**X**	
Controle de Orçamento (1 - 4)	**X**	
Controle de Orçamento (5 - 8)	**X**	
Controle de Orçamento (9 - 11)	**X**	
Financeiro		
Relatórios		

Tabela 7 - Demonstração dos casos de para refatoração do código que foram elaborados e implementados até o momento, separados por poucos validadores a serem implementados por cada módulo gerado.

Assim como afirmado por Sengupta (2004) e por Kniberg (2007), e após a realização de um questionário com os desenvolvedores da equipe de desenvolvimento, é possível avaliar e confirmar que tais requisitos são otimizados após utilização da técnica de TDD para desenvolvimento de um software:

- Maior compreensão de como desenvolver as telas;
- Maior compreensão sobre o que deve ser implementado;
- Maior agilidade no desenvolver das telas;
- Maior agilidade em incrementar requisitos as funcionalidades;
- Diminuição na quantidade de erros encontrados após a implementação e conseqüente aumento na qualidade do software*;*
- Maior pré-disposição em trabalhar com TDD em softwares futuros;

Pode-se observar também que a afirmação de Kniberg (2007) apresenta-se totalmente realista, pois após o desenvolvedor possuir um maior conhecimento de como trabalhar e utilizar a prática de TDD, o desenvolvimento do software tende a ser consideravelmente mais ágil e eficiente, porém desenvolvedores que nunca trabalharam com tal prática encontram grandes dificuldades em utilizar tal técnica, sendo esta uma desvantagem observada, visto que o início da utilização é muito complicada, mas após a obtenção de um conhecimento de utilização, dificilmente os desenvolvedores optaram por utilizar a metodologia convencional de desenvolvimento.

5 CONCLUSÃO

Após a realização deste trabalho pode-se concluir que ao ser utilizada a técnica de TDD, a equipe de desenvolvimento pode obter uma melhoria considerável no processo de desenvolvimento de software, visto que os desenvolvedores possuem uma melhor e mais otimizada compreensão a respeito do que deve ser implementado, há uma agilização no processo de desenvolvimento das telas do programa e também na refatoração do software que o capacita a obter um maior nível de completeza, dispondo também uma diminuição considerável na quantidade de erros encontrados, visto que a cada vez que o programa é incrementado, este objetiva validar alguns requisitos específicos até que o programa esteja completamente desenvolvido.

Observou-se também que a utilização da técnica por equipes que não possuem ao menos um profissional com experiência no utilizar da técnica, encontram grandes dificuldades para começarem a utilizá-la, visto que a atividade de elaboração dos casos de teste (principal atividade para aplicação do TDD), ser de difícil definição e implementação, ainda mais quando o desenvolvedor possui somente um conhecimento teórico. Porém, após uma obtenção de conhecimento de como a técnica pode ser utilizada, pode-se concluir que a técnica disponibiliza grandes vantagens em relação ao desenvolvimento convencional, tornando-se um considerável diferencial competitivo.

Baseando-se nas desvantagens (encontradas no início da utilização da técnica) e nas vantagens (apontadas acima) pode-se concluir que a utilização da técnica de TDD apresenta-se mais otimizada do que a metodologia convencional de desenvolvimento, visto que a equipe de desenvolvimento, a qual está utilizando a técnica (equipe do Laboratório Avançado de Software - LAS), já implementou utilizando ambas as metodologias, sendo observado um diferencial positivo na agilidade de implementação e no entendimento dos desenvolvedores sobre o que deve ser feito com a utilização do TDD.

Para realização de trabalhos futuros são propostas tais temáticas:

- Análise das vantagens e desvantagens da utilização da técnica TDD após conclusão do desenvolvimento do software de Cotação para Adecot;

- Análise da integração entre a técnica TDD para desenvolvimento de software e a ferramenta Mantis para gerenciamento de defeitos;
- Avaliação das vantagens e desvantagens da utilização da técnica TDD para o desenvolvimento distribuído de software;
- Avaliação das vantagens e desvantagens da integração entre a técnica TDD e a ferramenta Mantis para o desenvolvimento distribuído de software;
- Avaliar a diminuição na quantidade dos erros encontrados, com a utilização do TDD através de métricas de medição de software tais como: pontos por função, pontos por caso de uso e linhas de código;

Estudo empírico sobre a utilização da prática TDD em um ambiente acadêmico para o desenvolvimento de software em conjunto com a metodologia ágil SCRUM.

REFERÊNCIA

BANKI, A. L.; TANAKA, S. A. **Metodologias Ágeis: Uma Visão Prática.** Engenharia de Software Magazine. Rio de Janeiro, 4ª Edição. 2008.

BARBOSA, E. F.; MALDONADO, J. C.; VINCENZI, A. M. R.; DELAMARO, M. E. **Introdução ao Teste de Software.** 2007.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto for Agile Software Development.** 2001. Disponível em: <http://www.agilemanifesto.org/> . Acesso em: 10 de maio de 2010.

CALÇADO, V. L. X. de S. **Influência da Utilização de Processo Unificado, Testes e Métrica na Qualidade de Produtos de Software.** Brasília, 2007.

CAMPOS, F. M. **Qualidade, Qualidade de Software e Garantia de Qualidade de Software são as mesmas coisas?** 12 de março de 2008. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=1712>. Acesso em: 13 de maio de 2010.

CÓCARO, H.; LOPES, M. A.; CAMPOS, F. C. A. **Qualidade de Software Agropecuário: um estudo de caso.** Scielo, Lavras, 10 jun. 2005.

DAIBERT, M. S.; TOLEDO, J. V.; ARAÚJO, M. A. P. **Teste Automatizado de Software em Web Services.** Engenharia de Software Magazine. Rio de Janeiro, 8ª Edição. 2008.

FONSECA, I.; CAMPOS, A. **Por que Scrum?** Engenharia de Software Magazine. Rio de Janeiro, 4ª Edição. 2008.

FRANZEN, M. B.; BELLINI, C. G. P. **Arte ou Prática em Testes de Software?** REAd – 45º Edição, volume 11. 3 de maio-junho de 2005.

KNIBERG, H. **Scrum e XP direto das Trincheiras: Como Fazemos Scrum.** InfoQ. Série Desenvolvimento de Software Corporativo. 2007.

KOSCIANSKI, A.; SOARES, M. S. **Qualidade de Software: Aprenda as Metodologias e Técnicas mais Modernas para o Desenvolvimento de Software.** 2ª edição. São Paulo: Novatec Editora, 2007.

MAYER, D.; CHIARELLO, M. **Guia para Elaboração de Casos de Teste.** CELEPAR Informática do Paraná. 24 de Agosto de 2009.

NETO, A. C. D. **Introdução a Teste de Software.** Engenharia de Software Magazine**.** Rio de Janeiro, 1ª Edição. 2007.

NITA, E. de F. **Melhoria da Qualidade de Produto e de Processo de Software a partir da Análise de Indicadores de Teste.** Gramado, 2002.

MALDONADO, J. C.; BARBOSA, E. F.; VINCENZI, A. M. R.; DELAMARO, M. E.; SOUZA, S. R. S.; JINO, M.**Introdução ao Teste de Software (Versão 2004 - 01).** São Carlos, 2004.

PFLEEGER, S. L. **Engenharia de Software: Teoria e Prática.** 2º Edição. Tradução Dino Franklin; revisão técnica Ana Regina Cavalcanti da Rocha. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de Software.** 5º Edição. Rio de Janeiro: McGraw-Hill, 2002.

REZENDE, D. A. **Engenharia de Software e Sistemas de Informação.** 2ª Edição. Rio de Janeiro: Brasport, 2002.

SANTOS, R. F. **Scrum Experience.** Agosto de 2009.

SENGUPTA, B.; SINHA, V.; CHANDRA, S. **Test-Driven Global Software Development.** 2004.

SOARES, M. S. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software.** 2004.

SOMMERVILLE, I. **Engenharia do Software.** 6º Edição. Tradução André Maurício de Andrade Ribeiro; revisão técnica Kechi Hirama. São Paulo: Addison Wesley, 2003.

TORRES-ZENTENO, A. H.; MARTINS, E.; TORRES, R. S.; CUARESMA, M. J. E. **Teste de Desempenho em Aplicações SIG Web.** Unicamp, Campinas, 2005.

APÊNDICE – A

Neste apêndice serão apresentados os primeiros casos de teste elaborados, sendo este o modelo que objetiva a criação inicial das telas do software, tais telas são de: cadastro de produtos, cadastro de produtores, cadastro de fornecedores, cadastro de usuários, controle de orçamento e realização de pedidos.

CADASTRO DE PRODUTOS

CABEÇALHO

Caso de Teste: Cadastro de Produto.

Caso de Uso Relacionado: Cadastrar Produtos.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 11/09/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção produto.

Pós-condições: Produto deve ser inserido no banco de dados contendo todos os dados como: nome (Randap), categoria (herbicida), descrição (veneno para eliminar ervas daninhas), unidade (Lts, KG, PC, DZ) e o código do produto (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo produto;

2- Ator inicia o cadastramento dos produtos inserindo todos os dados do produto, tais como: nome, categoria, descrição e unidade de medida;

3- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

4- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Produto Cadastrado com Sucesso).

Resultado Esperado: Produto registrado no banco de dados sem causar conflito com qualquer outro dado que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE PRODUTORES

CABEÇALHO

Caso de Teste: Cadastro de Produtor.

Caso de Uso Relacionado: Cadastrar Produtores.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 18/09/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção produtor.

Pós-condições: Um novo produtor deve ser inserido no banco de dados contendo todos os dados como: nome (Saulo do Nascimento Machado), CPF (37520692876), Número Produtor (001), CadPro (0123456789), Sexo (Masculino), Data Nascimento (13/03/1989), Telefone Residencial (35421324), Telefone Comercial (35421324), Telefone Celular (99759655), E-mail (saulo.n.m@hotmail.com), Estado (Paraná), Cidade (Bandeirantes), Sitio (São Pedro), Logradouro (Rua Prefeito Rafael Antonacci n º 67), CEP (86360000), Bairro (Centro), Complemento (Apartamento 01) e o código do produtor (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo produtor;

2- Ator inicia o cadastramento do produtor inserindo todos os dados do produtor, tais como: nome, CPF, número produtor, CadPro, sexo, data

nascimento, telefone residencial, telefone comercial, telefone celular, e-mail, estado, cidade, sitio, logradouro, CEP, bairro, complemento;

3- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

4- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Produtor Cadastrado com Sucesso).

Resultado Esperado: Produtor registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE FORNECEDORES

CABEÇALHO

Caso de Teste: Cadastro de Fornecedor.

Caso de Uso Relacionado: Cadastrar Fornecedores.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 28/09/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção de fornecedor.

Pós-condições: Um novo fornecedor deve ser inserido no banco de dados contendo todos os dados como: nome (Domene), CNPJ (01.795.682/0001-39), Razão Social (Domene e Silvestre Ltda), Inscrição Estadual (9012968393), Telefone Comercial (44-3232-1353), Telefone fax (44-3232-1353), E-mail (domene@hotmail.com), Estado (Paraná), Cidade (Marialva), Logradouro (Rua Presidente Nereu Ramos n º 1.276), CEP (86990000), Bairro (Centro), Complemento (Fabrica) e o código do fornecedor (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo fornecedor;

2- Ator inicia o cadastramento do fornecedor inserindo todos os dados do fornecedor, tais como: nome, CNPJ, razão social, inscrição estadual, telefone comercial, telefone fax, e-mail, estado, cidade, logradouro, CEP, bairro, complemento;

3- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

4- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Fornecedor Cadastrado com Sucesso).

Resultado Esperado: Fornecedor registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE USUÁRIOS

CABEÇALHO

Caso de Teste: Cadastro de Usuário.

Caso de Uso Relacionado: Cadastrar Usuários.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 06/10/2010

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção usuário.

Pós-condições: Um novo usuário deve ser inserido no banco de dados contendo todos os dados como: nome (Saulo do Nascimento Machado), CPF (37520692876), *Login* (SauloNM), Senha (123321), Confirma Senha (123321); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo usuário;

2- Ator inicia o cadastramento do usuário inserindo todos os dados do usuário, tais como: nome, CPF, *login,* senha e confirmação da senha;

3- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

4- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Usuário Cadastrado com Sucesso).

Resultado Esperado: Usuário registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CONTROLE DE ORÇAMENTO

CABEÇALHO

Caso de Teste: Controle de Orçamento.

Caso de Uso Relacionado: Controlar Orçamento.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 10/10/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema e acessar a opção de orçamento. Anteriormente a esta execução é necessário que o usuário tenha cadastrado o fornecedor associado aos produtos orçados e ter cadastrado também os produtos relacionados para o orçamento.

Pós-condições: A cotação do preço de todos os produtos relacionados a um fornecedor deve ser registrada na base de dados, contendo uma relação entre o fornecedor (Domene), o ano de cotação (2010), um período do ano de cotação (2º semestre), o produto a ser orçado (Randap) e os valores do produto para dois meses (8,60) ou safra (9,90), estando estes dados disponíveis para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar e controlar o orçamento de alguns produtos;

2- Ator inicia o cadastramento do orçamento selecionando um fornecedor dos produtos cotados, preenchendo o ano do orçamento e selecionando o período do ano em que o orçamento foi realizado;

3- Ator seleciona o produto que foi orçado com o fornecedor informado, inserindo os valores levantados para pagamento no período de dois meses e o valor pago para a safra (seis meses);

4- Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir o novo produto orçado em uma tabela que será registrada na base de dados;

5- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

6- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Orçamento Cadastrado com Sucesso).

Resultado Esperado: Orçamento registrado no banco de dados relacionando o fornecedor com os produtos orçados e seus respectivos valores para a data de pagamento, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.

REALIZAÇÃO DE PEDIDOS

CABEÇALHO

Caso de Teste: Realização de Pedido.

Caso de Uso Relacionado: Realizar Pedido.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 27/10/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema e acessar a opção de pedido. Anteriormente a esta execução é necessário que o usuário tenha cadastrado o produtor o qual efetuará um pedido, o fornecedor dos produtos, os produtos orçados e ter cadastrado o orçamento dos produtos.

Pós-condições: O pedido de um produtor deve ser registrado na base de dados, contendo uma relação entre o nome do produtor (Saulo do Nascimento Machado), a data do pedido (25/10/2010), os produtos pedidos (tesoura - un), o fornecedor relacionado ao produto pedido (Domene), a opção de data de pagamento (dois meses ou safra com seus respectivos preços), a porcentagem da comissão (preenchida de padrão 5%), a quantidade pedida do respectivo produto (5), o numero do pedido (0001), um campo referente às observações e informações adicionais que podem ser inseridas pelo usuário, o campo valor total (250,00) e um código do pedido (que será auto-incrementado), estando estes dados disponíveis para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de realizar um pedido;

2- Ator inicia a realização de um pedido selecionando o produtor que efetuará o pedido e a data de realização do pedido;

3- Ator seleciona o produto a ser pedido e os fornecedores que disponibilizam este produto, passando a escolher a maneira de pagamento (2 meses ou safra), preencher a porcentagem da comissão, a quantidade de produtos, inserindo o numero do pedido e inserindo alguma observação ou informação adicional sobre o pedido realizado;

4- Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir um novo produto do pedido na tabela que será registrada na base de dados;

5- Sistema deve auto-incrementar o campo valor total e não disponibilizá-lo para alteração pelo ator;

6- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

7- O sistema deve exibir uma mensagem de confirmação da realização do pedido (Pedido Realizado com Sucesso).

Resultado Esperado: Pedido registrado no banco de dados relacionando o produtor com data de pedido, os produtos do pedido e seus respectivos fornecedores, selecionando a data de pagamento e seus valores, com o valor da comissão, a quantidade de cada produto, o numero do pedido e o valor total do

pedido realizado, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.

APÊNDICE – B

Neste apêndice serão apresentados os casos de teste elaborados para o segundo modelo utilizado, com o objetivo de possibilitar o planejamento dos testes a serem realizados pelo analista de teste e auxiliar na refatoração do software, de acordo com a necessidade de validação específica de cada funcionalidade.

CADASTRO DE PRODUTOS

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

   Exemplo: <objeto selecionado; TAB; resultado esperado>

   Caso 1 <NOVO; TAB; CONSULTAR>

   Caso 2 <SAIR; TAB; Nome>

   Caso 3 <Nome; TAB; Categoria>

   Caso 4 <Unidade; TAB; NOVO>

2. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

   Exemplo: <campo; mensagem de duplicidade>

   Caso 1 <Nome; Produto já cadastrado>

3. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

   Exemplo: <quantidade (nome do campo); resultado esperado>

   Caso 1 <100 (Nome); Correto>

   Caso 2 <101 (Nome); Erro>

   Caso 3 <70 (Categoria); Correto>

   Caso 4 <71 (Categoria); Erro>

   Caso 5 <150 (Descrição); Correto>

   Caso 6 <151 (Descrição); Erro>

   Caso 7 <10 (Unidade); Correto >

   Caso 8 <11 (Unidade); Erro>

4. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

   Exemplo: <campo sem nada; resultado esperado>

   Caso 1 <Nome; Erro>

   Caso 2 <Categoria; Erro>

   Caso 3 <Descrição; Correto>

   Caso 4 <Unidade; Erro>

5. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

   Exemplo: <campo; mensagem de auxílio>

   Caso 1 <Nome; Insira o nome do produto. Ex: Randap>

   Caso 2 <Categoria; Inserir a categoria o produto. Ex: Herbicida>

   Caso 3 <Descrição; Insira informações que caracterizem o produto.>

   Caso 4 <Unidade; Informe a unidade de medida do produto. EX: Lts (litros), Pc (pacote), Kg>

6. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

   Exemplo: <botão; estado; mesagem>

   Caso 1 <SALVAR; correto; Produto cadastrado com sucesso>

   Caso 2 <SALVAR; errado; Produto já cadastrado>

   Caso 3 <SAIR; correto; Cadastro não salvo. Deseja sair desta página?>

   Caso 4 <SAIR; errado; Cadastro não salvo. Deseja sair desta página?>

CADASTRO DE PRODUTORES

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

   Exemplo: <objeto selecionado; TAB; resultado esperado>

   Caso 1 <NOVO; TAB; CONSULTAR>

   Caso 2 <SAIR; TAB; Nome>

   Caso 3 <Nome; TAB; CPF>

   Caso 4 <Complemento; TAB; NOVO>

2. **Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas).**

   Exemplo: <campo; dado de entrada; resultado esperado>

   Caso 1 <CPF; Saulo 123 Machado; Mensagem de erro>

   Caso 2 <Data de Nascimento; hoje; Mensagem de erro>

   Caso 3 <Data de Nascimento; 10/15/2050; Mensagem de erro>

   Caso 4 <Telefone residencial; não tem; Mensagem de erro>

   Caso 5 <Telefone comercial; não tem; Mensagem de erro>

   Caso 6 <Telefone celular; não tem; Mensagem de erro>

   Caso 7 <CEP; não sei; Mensagem de erro>

3. **Teste: Validação das mensagens para erro de preenchimento dos campos.**

   Exemplo: <campo; mensagem de erro>

   Caso 1 <CPF; Insira somente números Ex. 123 456 789 00>

   Caso 2 <Data de nascimento; Insira uma data valida Ex. 10/10/2010>

   Caso 3 <Telefone residencial; Insira somente números Ex. 43 3542 1000>

   Caso 4 <Telefone comercial; Insira somente números Ex. 43 3542 1000>

   Caso 5 <Telefone celular; Insira somente números Ex. 43 9977 7799>

   Caso 6 <CEP; Insira somente números Ex. 86360000>

4. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

   Exemplo: <campo; mensagem de duplicidade>

   Caso 1 <Nome; Produtor já cadastrado>

   Caso 2 <CPF; CPF já cadastrado>

   Caso 3 <CadPro; CAdPro já cadastrado>

5. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

   Exemplo: <quantidade (nome do campo); resultado esperado>

   Caso 1 <120 (Nome); Correto>

   Caso 2 <121 (Nome); Erro>

   Caso 3 <11 (CPF); Correto>

   Caso 4 <12 (CPF); Erro>

   Caso 5 <9 (CPF); Erro>

   Caso 6 <8 (Data); Correto>

Caso 7 <10 (Data); Erro>

Caso 8 <4 (Data); Erro>

Caso 9 <10 (Telefone residencial); Correto>

Caso 10 <11 (Telefone residencial); Erro>

Caso 11 <10 (Telefone comercial); Correto>

Caso 12 <11 (Telefone comercial); Erro>

Caso 13 <10 (Telefone celular); Correto>

Caso 14 <11 (Telefone celular); Erro>

Caso 15 <100 (e-mail); Correto>

Caso 16 <101 (e-mail); erro>

Caso 17 <80 (sitio); correto>

Caso 18 <81 (sitio); erro>

Caso 19 <120 (logradouro); correto>

Caso 20 <121 (logradouro); erro>

Caso 21 <9 (CEP); correto>

Caso 22 <10 (CEP); erro>

Caso 23 <100 (Bairro); correto>

Caso 24 <101 (bairro); erro>

Caso 25 <100 (complemento); correto>

Caso 26 <101 (complemento); erro>

6. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

Exemplo: <campo sem nada; resultado esperado>

Caso 1 <Nome; Erro>

Caso 2 <CPF; Erro>

Caso 3 <Número Produtor; Correto>

Caso 4 <CadPro; correto>

Caso 5 <Sexo; Erro>

Caso 6 <Data nascimento; erro>

Caso 7 <Telefone residencial; correto>

Caso 8 <Telefone comercial; correto>

Caso 9 <Telefone celular; correto>

Caso 10 <e-mail; correto>

Caso 11 <Estado; Erro>

Caso 12 <Cidade; Erro>

Caso 13 <Sítio; correto>

Caso 14 <Logradouro; correto>

Caso 15 <CEP; erro>

Caso 16 <Bairro; erro>

Caso 17 <Complemento; correto>

7. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

Exemplo: <campo; mensagem de auxílio>

Caso 1 <Nome; Insira o nome do produtor. Ex: José Maria>

Caso 2 <CPF; Insira um CPF valido. Ex: 123 456 789 00>

Caso 3 <Número Produtor; Insira o numero do produtor. Ex: 0001>

Caso 4 <CadPro; Insira o numero de cadastro de produtor rural. Ex: 0150 >

Caso 5 <Sexo; Selecione o sexo do produtor. Ex: Masculino>

Caso 6 <Data nascimento; Insira uma data existente. Ex: 10/10/1990>

Caso 7 <Telefone residencial; Insira um numero do telefone residencial. Ex: 43 3542 1000>

Caso 8 <Telefone comercial; Insira um numero do telefone comercial. Ex: 43 3542 2000>

Caso 9 <Telefone celular; Insira um numero do telefone celular. Ex: 43 9975 1000>

Caso 10 <e-mail; Insira um e-mail. Ex: produtor@hotmail.com>

Caso 11 <Estado; Selecione o estado onde o produtor mora. Ex: Paraná>

Caso 12 <Cidade; Selecione a cidade onde o produtor mora. Ex: Bandeirantes>

Caso 13 <Sítio; Informe o sítio onde o produtor mora. Ex: São Pedro>

Caso 14 <Logradouro; Informe o endereço em que o produtor mora. Ex: R: São Paulo nº 150 >

Caso 15 <CEP; Informe o CEP da cidade em que o produtor mora. Ex: 86360 000 >

Caso 16 <Bairro; Informe o bairro do produtor. Ex: Vila Maria>

Caso 17 <Complemento; Informe alguma característica da casa do produtor. Ex: Sobrado>

8. **Teste: Validar a veracidade do CPF.**

   Exemplo: <dado de entrada; resultado esperado>

   Caso 1 <CPF inexistente; Mensagem de erro: CPF Inexistente>

   Caso 2 <CPF já cadastrado; Mensagem de erro: CPF Já Cadastrado>

   Caso 3 <CPF não preenchido; Mensagem de erro: Campo obrigatório>

   Caso 4 <CPF válido; Correto>

9. **Teste: Validar a data de nascimento, com datas existentes.**

   Exemplo: <dado de entrada; resultado esperado>

   Caso 1 <Data que ainda não chegou; Mensagem de erro: Data Inexistente>

   Caso 2 <Data incompleta; Mensagem de erro: Data Inexistente>

   Caso 3 <Data correta; Correto>

10. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

    Exemplo: <botão; estado; mesagem>

    Caso 1 <SALVAR; correto; Produtor cadastrado com sucesso>

    Caso 2 <SALVAR; errado; Produtor já cadastrado>

    Caso 3 <SAIR; correto; Cadastro não salvo. Deseja sair desta página?>

    Caso 4 <SAIR; errado; Cadastro não salvo. Deseja sair desta página?>

CADASTRO DE FORNECEDORES

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

   Exemplo: <objeto selecionado; TAB; resultado esperado>

   Caso 1 <NOVO; TAB; CONSULTAR>

   Caso 2 <SAIR; TAB; Nome>

   Caso 3 <Nome; TAB; CNPJ>

   Caso 4 <Complemento; TAB; NOVO>

2. **Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas).**

   Exemplo: <campo; dado de entrada; resultado esperado>

   Caso 1 <CNPJ; 123 Saulo 123; Mensagem de erro>

Caso 2 <Inscrição Estadual; a mesma; Mensagem de erro>

Caso 3 <Telefone comercial; não tem; Mensagem de erro>

Caso 4 <Telefone fax; não tem; Mensagem de erro>

Caso 5 <CEP; não sei; Mensagem de erro>

3. **Teste: Validação das mensagens para erro de preenchimento dos campos.**

   Exemplo: <campo; mensagem de erro>

   Caso 1 <CNPJ; Insira somente números. Ex. 01 795 682 0001 39>

   Caso 2 <Inscrição Estadual; Insira somente números. Ex. 1234567890>

   Caso 3 <Telefone comercial; Insira somente números. Ex. 44 3232 1353>

   Caso 4 <Telefone fax; Insira somente números. Ex. 44 3232 1353>

   Caso 6 <CEP; Insira somente números. Ex. 86990 000>

4. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

   Exemplo: <campo; mensagem de duplicidade>

   Caso 1 <Nome; Fornecedor já cadastrado>

   Caso 2 <CNPJ; CNPJ já cadastrado>

   Caso 3 <Inscrição Estadual; Inscrição Estadual Já Cadastrada>

5. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

   Exemplo: <quantidade (nome do campo); resultado esperado>

   Caso 1 <100 (Nome); Correto>

   Caso 2 <101 (Nome); Erro>

   Caso 3 <14 (CNPJ); Correto>

   Caso 4 <15 (CNPJ); Erro>

   Caso 5 < 10 (CNPJ) Erro>

   Caso 6 <101 (Razão Social); Erro>

   Caso 7 < 100 (Razão Social); Correto>

   Caso 8 <10 (Inscrição Estadual); Correto>

   Caso 9 <12 (Inscrição Estadual); Erro>

   Caso 10 <10 (Telefone comercial); Correto>

   Caso 11 <11 (Telefone comercial); Erro>

   Caso 12 <10 (Telefone fax); Correto>

   Caso 13 <11 (Telefone fax); Erro>

Caso 14 <100 (e-mail); Correto>

Caso 15 <101 (e-mail); erro>

Caso 16 <120 (logradouro); correto>

Caso 17 <121 (logradouro); erro>

Caso 18 <9 (CEP); correto>

Caso 19 <10 (CEP); erro>

Caso 20 <100 (Bairro); correto>

Caso 21 <101 (bairro); erro>

Caso 22 <100 (complemento); correto>

Caso 23 <101 (complemento); erro>

6. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

   Exemplo: <campo sem nada; resultado esperado>

   Caso 1 <Nome; Erro>

   Caso 2 <CNPJ; Erro>

   Caso 3 <Razão Social; Erro>

   Caso 4 <Inscrição Estadual; Erro>

   Caso 5 <Telefone comercial; correto>

   Caso 6 <Telefone fax; correto>

   Caso 7 <e-mail; correto>

   Caso 8 <Estado; Erro>

   Caso 9 <Cidade; Erro>

   Caso 10 <Logradouro; correto>

   Caso 11 <CEP; erro>

   Caso 12 <Bairro; erro>

   Caso 13 <Complemento; correto>

7. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

   Exemplo: <campo; mensagem de auxílio>

   Caso 1 <Nome; Insira o nome do fornecedor. Ex: Domene Agropecuária>

   Caso 2 <CNPJ; Insira um CNPJ valido. Ex: 01 795 682 001 39>

   Caso 3 <Razão Social; Insira a razão social do fornecedor. Ex: Domene & Silvestre LTDA>

Caso 4 < Inscrição Estadual; Insira o numero da inscrição estadual. Ex: 90129683 93 >

Caso 5 <Telefone comercial; Insira um numero do telefone comercial. Ex: 44 3232 1353>

Caso 6 <Telefone fax; Insira um numero do fax. Ex: 44 3232 1353>

Caso 7 <e-mail; Insira um e-mail. Ex: domenesilvestre@hotmail.com>

Caso 8 <Estado; Selecione o estado do fornecedor. Ex: Paraná>

Caso 9 <Cidade; Selecione a cidade do fornecedor. Ex: Marialva>

Caso 10 <Logradouro; Informe o endereço do fornecedor. Ex: R: Pres. Nereu Ramos, 1276 >

Caso 11<CEP; Informe CEP da cidade do fornecedor. Ex: 86990000 >

Caso 12 <Bairro; Informe o bairro do produtor. Ex: Centro>

Caso 13 <Complemento; Informe alguma característica da casa do produtor. Ex: Fábrica>

8. **Teste: Validar a veracidade do CNPJ.**

   Exemplo: <dado de entrada; resultado esperado>

   Caso 1 <CNPJ inexistente; Mensagem de erro: CNPJ Inexistente>

   Caso 2 <CNPJ já cadastrado; Mensagem de erro: CNPJ Já Cadastrado>

   Caso 3 <CNPJ não preenchido; Mensagem de erro: Campo obrigatório>

   Caso 4 <CNPJ válido; Correto>

9. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

   Exemplo: <botão; estado; mesagem>

   Caso 1 <SALVAR; correto; Fornecedor Cadastrado com Sucesso>

   Caso 2 <SALVAR; errado; Fornecedor Já Cadastrado>

   Caso 3 <SAIR; correto; Cadastro não salvo. Deseja sair desta página?>

   Caso 4 <SAIR; errado; Cadastro não salvo. Deseja sair desta página?>

CADASTRO DE USUÁRIOS

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

Exemplo: <objeto selecionado; TAB; resultado esperado>

Caso 1 <NOVO; TAB; CONSULTAR>

Caso 2 <SAIR; TAB; Nome>

Caso 3 <Nome; TAB; CPF>

Caso 4 <Confirma Senha; TAB; NOVO>

2. **Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas).**

Exemplo: <campo; dado de entrada; resultado esperado>

Caso 1 <CPF; 123CPF321; Mensagem de erro>

Caso 2 <Confirma Senha; Dado diferente da senha informada; Mensagem de Erro>

3. **Teste: Validação das mensagens para erro de preenchimento dos campos.**

Exemplo: <campo; mensagem de erro>

Caso 1 <CPF; Insira somente números. Ex. 375 206 928 76>

Caso 2 <Confirma Senha; Senha errada. Insira a mesma senha que foi informada acima>

4. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

Exemplo: <campo; mensagem de duplicidade>

Caso 1 <Nome; Usuário já cadastrado>

Caso 2 <CPF; CPF já cadastrado>

5. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

Exemplo: <quantidade (nome do campo); resultado esperado>

Caso 1 <100 (Nome); Correto>

Caso 2 <101 (Nome); Erro>

Caso 3 <14 (CPF); Correto>

Caso 4 <15 (CPF); Erro>

Caso 5 < 10 (CPF) Erro>

Caso 6 <201 (Login); Erro>

Caso 7 < 200 (Login); Correto>

Caso 8 <10 (Senha); Correto>

Caso 9 <11 (Senha); Erro>

Caso 10 <10 (Confirma Senha); Correto>

Caso 11 <11 (Confirma Senha); Erro>

6. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

   Exemplo: <campo sem nada; resultado esperado>

   Caso 1 <Nome; Erro>

   Caso 2 <CPF; Erro>

   Caso 3 <Login; Erro>

   Caso 4 <Senha; Erro>

   Caso 5 <Confirma Senha; Erro>

7. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

   Exemplo: <campo; mensagem de auxílio>

   Caso 1 <Nome; Insira o nome do usuário. Ex: José Maria>

   Caso 2 <CPF; Insira um CPF valido. Ex: 123 456 789 00>

   Caso 3 <Login; Insira o nome que será digitado para acessar o programa. Ex. zemaria>

   Caso 4 < Senha; Insira a senha para acessar o programa com seu login. Ex: adecot>

   Caso 5 <Confirma Senha; Confirma a senha informada no campo acima. Ex:adecot>

8. **Teste: Validar a veracidade do CPF.**

   Exemplo: <dado de entrada; resultado esperado>

   Caso 1 <CPF inexistente; Mensagem de erro: CPF Inexistente>

   Caso 2 <CPF já cadastrado; Mensagem de erro: CPF Já Cadastrado>

   Caso 3 <CPF não preenchido; Mensagem de erro: Campo obrigatório>

   Caso 4 <CPF válido; Correto>

9. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

   Exemplo: <botão; estado; mesagem>

   Caso 1 <SALVAR; correto; Usuário Cadastrado com Sucesso>

   Caso 2 <SALVAR; errado; Usuário Já Cadastrado>

   Caso 3 <SAIR; correto; Cadastro não salvo. Deseja sair desta página?>

Caso 4 <SAIR; errado; Cadastro não salvo. Deseja sair desta página?>

CONTROLE DE ORÇAMENTO

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

   Exemplo: <objeto selecionado; TAB; resultado esperado>

   Caso 1 <NOVO; TAB; CONSULTAR>

   Caso 2 <SAIR; TAB; Fornecedor>

   Caso 3 <Semestre; TAB; OK>

   Caso 4 <OK; TAB; Produto>

   Caso 5 <Safra; TAB; ADICIONAR>

   Caso 6 <Ultimo item da tabela; TAB; NOVO>

2. **Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas).**

   Exemplo: <campo; dado de entrada; resultado esperado>

   Caso 1 <Ano; Atual; Mensagem de erro>

   Caso 2 <2 Meses; 10 reais; Mensagem de erro>

   Caso 3 <Safra; 12 reais; Mensagem de erro>

3. **Teste: Validação das mensagens para erro de preenchimento dos campos.**

   Exemplo: <campo; mensagem de erro>

   Caso 1 <Ano; Insira somente números. Ex. 2010>

   Caso 2 <2 Meses; Insira somente números. Ex: 10 00>

   Caso 3 <Safra; Insira somente números. Ex: 12 00>

4. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

   Exemplo: <campo; mensagem de duplicidade>

   Caso 1 <Fornecedor, Ano e Semestre; Orçamento já cadastrado>

5. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

   Exemplo: <quantidade (nome do campo); resultado esperado>

   Caso 1 <4 (Ano); Correto>

   Caso 2 <5 (Ano); Erro>

   Caso 3 <3 (Ano); erro>

Caso 4 <7 (2 Meses); Correto>

Caso 5 <8 (2 Meses); Erro>

Caso 6 < 7 (Safra); Correto>

Caso 7 <8 (Safra); Erro>

6. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

   Exemplo: <campo sem nada; resultado esperado>

   Caso 1 <Fornecedor; Erro>

   Caso 2 <Ano; Erro>

   Caso 3 <Semestre; Erro>

   Caso 4 <Mercadoria; Erro>

   Caso 5 <2 Meses; Erro>

   Caso 6 <Safra; Erro>

7. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

   Exemplo: <campo; mensagem de auxílio>

   Caso 1 <Fornecedor; Selecione o fornecedor dos produtos orçados. Ex: Domene>

   Caso 2 <Ano; Insira um ano valido. Ex: 2010>

   Caso 3 <Semestre; Selecione o semestre em que o orçamento foi realizado. Ex 2º Semestre>

   Caso 4 < Mercadoria; Selecione a mercadoria para inserção dos valores. Ex: Herbicida>

   Caso 5 <2 Meses; Insira qual o valor da mercadoria para ser paga em 2 meses. Ex: 10 00>

   Caso 6 <Safra; Insira o valor da mercadoria para ser paga na safra. Ex: 12 00>

8. **Teste: Validar a não inserção de valor igual a zero.**

   Exemplo: <campo; Mensagem de erro>

   Caso 1 <Ano; Insira um ano diferente de zero>

   Caso 2 <2 Meses; Insira um valor diferente de zero>

   Caso 3 <Safra; Insira um valor diferente de zero>

9. **Teste: Validar o ano do orçamento.**

   Exemplo: <dado de entrada; resultado esperado>

   Caso 1 <Ano que ainda não chegou; Mensagem de erro: Ano Inexistente>

   Caso 2 <Ano incompleto; Mensagem de erro: Ano Inexistente>

   Caso 3 <Ano correto; Correto>

10. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

    Exemplo: <botão; estado; mesagem>

    Caso 1 <SALVAR; correto; Orçamento Cadastrado com Sucesso>

    Caso 2 <OK; errado; Orçamento Já Cadastrado>

    Caso 3 <SAIR; correto; Orçamento não salvo. Deseja sair desta página?>

    Caso 4 <SAIR; errado; Orçamento não salvo. Deseja sair desta página?>

11. **Teste: Validar a quantidade de dados que a tabela suporta.**

    Exemplo: <quantidade; resultado esperado>

    Caso 1 <5; primeira tabela>

    Caso 2 <6; segunda tabela>

    Caso 3 <11; terceira tabela>

REALIZAÇÃO DE PEDIDOS

Casos de Teste para validação:

1. **Teste: Seqüência de alternância de campos utilizando a tecla TAB.**

   Exemplo: <objeto selecionado; TAB; resultado esperado>

   Caso 1 <NOVO; TAB; CONSULTAR>

   Caso 2 <SAIR; TAB; Produtor>

   Caso 3 <Data; TAB; OK>

   Caso 4 <OK; TAB; Produto>

   Caso 5 <Observação; TAB; ADICIONAR>

   Caso 6 <Valor Total; TAB; NOVO>

2. **Teste: Verificação da inserção correta dos dados, relacionados ao tipo que os campos esperam (numérico, letras, datas).**

Exemplo: <campo; dado de entrada; resultado esperado>

Caso 1 <Data; Atual; Mensagem de erro>

Caso 2 <Quantidade; dez; Mensagem de erro>

Caso 3 <Número Pedido; cem; Mensagem de erro>

Caso 4 <Comissão; 10 reais; Mensagem de erro>

3. **Teste: Validação das mensagens para erro de preenchimento dos campos.**

Exemplo: <campo; mensagem de erro>

Caso 1 <Data; Insira uma data válida. Ex. 10/10/2010>

Caso 2 <Quantidade; Insira somente números. Ex: 10>

Caso 3 <Número Pedido; Insira somente números. Ex: 120>

Caso 4 <Comissão; Insira somente números. EX: 5>

4. **Teste: Validação da inexistência do novo dado a ser cadastrado para que não haja duplicidade de dados na Base de Dados.**

Exemplo: <campo; mensagem de duplicidade>

Caso 1 <Número Pedido; Pedido já cadastrado>

5. **Teste: Verificação do numero limite de caracteres em todos os campos com relação à Base de Dados.**

Exemplo: <quantidade (nome do campo); resultado esperado>

Caso 1 <8 (Data); Correto>

Caso 2 <10 (Data); Erro>

Caso 3 <5 (Data); erro>

Caso 4 <3 (Quantidade); Correto>

Caso 5 <4 (Quantidade); Erro>

Caso 6 < 6 (Número Pedido); Correto>

Caso 7 <7 (Número Pedido); Erro>

Caso 8 <2 (Comissão); Correto>

Caso 9 <5 (Comissão); Erro>

6. **Teste: Tentar cadastrar campos em branco para verificação da obrigatoriedade dos campos requeridos.**

Exemplo: <campo sem nada; resultado esperado>

Caso 1 <Produtor; Erro>

Caso 2 <Data; Erro>

Caso 3 <Produto; Erro>

Caso 4 <Fornecedor; Erro>

Caso 5 <2 Meses; Erro>

OR

Caso 6 <Safra; Erro>

Caso 7 <Quantidade; Erro>

Caso 8 <Número Pedido; Erro>

Caso 9 <Observação; Correto>

7. **Teste: Validação da existência de TITLES de auxílio em todos os campos.**

Exemplo: <campo; mensagem de auxílio>

Caso 1 <Produtor; Selecione o produtor que realizou o pedido. Ex: José Maria>

Caso 2 <Data; Insira uma data valida. Ex: 10/10/2010>

Caso 3 <Produto; Selecione o produto pedido. Ex: Randap>

Caso 4 < Fornecedor; Selecione o fornecedor do produto pedido. Ex: Domene>

Caso 5 <2 Meses; Valor do produto para ser paga em 2 meses. Ex: 10 00>

Caso 6 <Safra; Valor do produto para ser paga na safra. Ex: 12 00>

Caso 7 <Comissão; Insira a porcentagem de comissão deste pedido. Ex: 5>

Caso 8 <Quantidade; Insira a quantidade pedida do produto. Ex: 3>

Caso 9 <Número Pedido; Insira qual o número do pedido. Ex: 000150>

Caso 10 <Observação; Insira alguma informação especifica do pedido. Ex: Urgência>

8. **Teste: Validar a não inserção de valor igual a zero.**

Exemplo: <campo; Mensagem de erro>

Caso 1 <Data; Insira uma data valida>

Caso 2 <Quantidade; Insira uma quantidade diferente de zero>

Caso 3 <Número Pedido; Insira um número de pedido diferente de zero>

9. **Teste: Validar o ano do orçamento.**

Exemplo: <dado de entrada; resultado esperado>

Caso 1 <Data que ainda não chegou; Mensagem de erro: Data Inexistente>

Caso 2 <Data incompleta; Mensagem de erro: Data Inexistente>

Caso 3 <Data correta; Correto>

10. **Teste: Validação dos pop-ups de confirmação para as operações requeridas.**

Exemplo: <botão; estado; mesagem>

Caso 1 <SALVAR; correto; Pedido Concluído Não Pode Ser Alterado. Deseja Realizar o Pedido?>

Caso 2 <OK; correto; Pedido Realizado com Sucesso>

Caso 3 <SAIR; correto; Pedido não salvo. Deseja sair desta página?>

Caso 4 <SAIR; errado; Pedido não salvo. Deseja sair desta página?>

11. **Teste: Validar a quantidade de dados que a tabela suporta.**

Exemplo: <quantidade; resultado esperado>

Caso 1 <5; primeira tabela>

Caso 2 <6; segunda tabela>

Caso 3 <11; terceira tabela>

APÊNDICE – C

Neste apêndice serão apresentados os casos de testes elaborados para a integração entre os dois modelos gerados anteriormente. Este modelo possui o objetivo de subsidiar a refatoração do software, apoiado pelas validações que foram planejadas no segundo modelo elaborado, conforme as especificações de cada funcionalidade.

Os modelos apresentados possuem todas as validações a serem incrementadas em cada funcionalidade, porém o requerimento de implementação para cada tela foi entregue aos desenvolvedores de acordo como citado na tabela 7, contendo entre 3 a 4 incrementos por cada caso de teste a ser implementado.

CADASTRO DE PRODUTOS

CABEÇALHO

Caso de Teste: Cadastro de Produto.

Caso de Uso Relacionado: Cadastrar Produtos.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 03/11/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção produto.

Pós-condições: Produto deve ser inserido no banco de dados contendo todos os dados como: nome (Randap), categoria (herbicida), descrição (veneno para eliminar ervas daninhas), unidade (Lts, KG, PC, DZ) e o código do produto (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo produto;

2- Ator inicia o cadastramento dos produtos inserindo todos os dados do produto, tais como: nome, categoria, descrição e unidade de medida;

3- Sistema deve garantir a correta alternância dos campos e dos botões com a utilização da tecla TAB;

4- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados com algum dado do Banco de Dados;

5- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

6- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

7- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

8- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

9- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

10- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Produto Cadastrado com Sucesso).

Resultado Esperado: Produto registrado no banco de dados sem causar conflito com qualquer outro dado que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE PRODUTORES

CABEÇALHO

Caso de Teste: Cadastro de Produtor.

Caso de Uso Relacionado: Cadastrar Produtores.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 05/11/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção produtor.

Pós-condições: Um novo produtor deve ser inserido no banco de dados contendo todos os dados como: nome (Saulo do Nascimento Machado), CPF (37520692876), Número Produtor (001), CadPro (0123456789), Sexo (Masculino), Data Nascimento (13/03/1989), Telefone Residencial (35421324), Telefone Comercial (35421324), Telefone Celular (99759655), E-mail (saulo.n.m@hotmail.com), Estado (Paraná), Cidade (Bandeirantes), Sitio (São Pedro), Logradouro (Rua Prefeito Rafael Antonacci n º 67), CEP (86360000), Bairro (Centro), Complemento (Apartamento 01) e o código do produtor (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo produtor;

2- Ator inicia o cadastramento do produtor inserindo todos os dados do produtor, tais como: nome, CPF, número produtor, CadPro, sexo, data nascimento, telefone residencial, telefone comercial, telefone celular, e-mail, estado, cidade, sitio, logradouro, CEP, bairro, complemento;

3- Sistema deve garantir a correta alternância dos campos e dos botões com a utilização da tecla TAB;

4- Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data);

5- Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos;

6- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados com algum dado do Banco de Dados;

7- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

8- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

9- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

10- Sistema deve validar a veracidade do CPF que for informado no cadastro;

11- Sistema deve validar a inserção de uma correta data de nascimento;

12- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

13- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

14- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Produtor Cadastrado com Sucesso).

Resultado Esperado: Produtor registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE FORNECEDORES

CABEÇALHO

Caso de Teste: Cadastro de Fornecedor.

Caso de Uso Relacionado: Cadastrar Fornecedores.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 09/11/2010

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção de fornecedor.

Pós-condições: Um novo fornecedor deve ser inserido no banco de dados contendo todos os dados como: nome (Domene), CNPJ (01.795.682/0001-39), Razão Social (Domene e Silvestre Ltda), Inscrição Estadual (9012968393), Telefone

Comercial (44-3232-1353), Telefone fax (44-3232-1353), E-mail (domene@hotmail.com), Estado (Paraná), Cidade (Marialva), Logradouro (Rua Presidente Nereu Ramos n º 1.276), CEP (86990000), Bairro (Centro), Complemento (Fabrica) e o código do fornecedor (que será auto-incrementado); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo fornecedor;

2- Ator inicia o cadastramento do fornecedor inserindo todos os dados do fornecedor, tais como: nome, CNPJ, razão social, inscrição estadual, telefone comercial, telefone fax, e-mail, estado, cidade, logradouro, CEP, bairro, complemento;

3- Sistema deve garantir a correta alternância dos campos e dos botões com a utilização da tecla TAB;

4- Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data);

5- Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos;

6- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados com algum dado do Banco de Dados;

7- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

8- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

9- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

10- Sistema deve validar a veracidade do CNPJ que for informado no cadastro;

11- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

12- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

13- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Fornecedor Cadastrado com Sucesso).

Resultado Esperado: Fornecedor registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CADASTRO DE USUÁRIOS

CABEÇALHO

Caso de Teste: Cadastro de Usuário.

Caso de Uso Relacionado: Cadastrar Usuários.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 11/11/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema; ter acessado a opção de cadastros na tela inicial e selecionar a opção usuário.

Pós-condições: Um novo usuário deve ser inserido no banco de dados contendo todos os dados como: nome (Saulo do Nascimento Machado), CPF (37520692876), *Login* (SauloNM), Senha (123321); assim estando disponível para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar um novo usuário;

2- Ator inicia o cadastramento do usuário inserindo todos os dados do usuário, tais como: nome, CPF, *login,* senha;

3- Sistema deve garantir a correta alternância dos campos e dos botões com a utilização da tecla TAB;

4- Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data);

5- Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos;

6- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados com algum dado do Banco de Dados;

7- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

8- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

9- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

10- Sistema deve validar a veracidade do CPF que for informado no cadastro;

11- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

12- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

13- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Usuário Cadastrado com Sucesso).

Resultado Esperado: Usuário registrado no banco de dados sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados e estar disponível para realização de consultas e alterações dos dados registrados.

CONTROLE DE ORÇAMENTO

CABEÇALHO

Caso de Teste: Controle de Orçamento.

Caso de Uso Relacionado: Controlar Orçamento.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 11/11/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema e acessar a opção de orçamento. Anteriormente a esta execução é necessário que o usuário tenha cadastrado o fornecedor associado aos produtos orçados e ter cadastrado também os produtos relacionados para o orçamento.

Pós-condições: A cotação do preço de todos os produtos relacionados a um fornecedor deve ser registrada na base de dados, contendo uma relação entre o fornecedor (Domene), o ano de cotação (2010), um período do ano de cotação (2º semestre), o produto a ser orçado (Randap) e os valores do produto para dois meses (8,60) ou safra (9,90), estando estes dados disponíveis para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de cadastrar e controlar o orçamento de alguns produtos;

2- Ator inicia o cadastramento do orçamento selecionando um fornecedor dos produtos cotados, preenchendo o ano do orçamento e selecionando o período do ano em que o orçamento foi realizado;

3- Ator seleciona o produto que foi orçado com o fornecedor informando, inserindo os valores levantados para pagamento no período de dois meses e o valor pago para a safra (seis meses);

4- Sistema deve garantir a correta alternância dos campos com a utilização da tecla TAB;

5- Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data);

6- Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos;

7- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados;

8- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

9- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

10- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

11- Sistema deve validar a não inserção dos valores igual a zero para os campos Ano, 2 meses e Safra;

12- Sistema deve validar a inserção de um ano valido para o correto cadastramento;

13- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

14- Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir o novo produto orçado em uma tabela que será registrada na base de dados;

15- Sistema deve validar a correta inserção dos dados na tabela, para que não haja perda de qualquer dado;

16- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

17- O sistema deve exibir uma mensagem de confirmação da realização do processo de cadastro (Orçamento Cadastrado com Sucesso).

Resultado Esperado: Orçamento registrado no banco de dados relacionando o fornecedor com os produtos orçados e seus respectivos valores para a data de pagamento, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.

REALIZAÇÃO DE PEDIDOS

CABEÇALHO

Caso de Teste: Realização de Pedido.

Caso de Uso Relacionado: Realizar Pedido.

Responsável: Analista de teste – Saulo; Programador – André.

Data: 15/11/2010.

Tipo de Teste: Teste de Validação.

Abordagem: Realização de Teste Manualmente.

Pré-condições: Usuário deve estar *Logado* no sistema e acessar a opção de pedido. Anteriormente a esta execução é necessário que o usuário tenha cadastrado o produtor o qual efetuará um pedido, o fornecedor dos produtos, os produtos orçados e ter cadastrado o orçamento dos produtos.

Pós-condições: O pedido de um produtor deve ser registrado na base de dados, contendo uma relação entre o nome do produtor (Saulo do Nascimento Machado), a data do pedido (25/10/2010), os produtos pedidos (tesoura - un), o fornecedor relacionado ao produto pedido (Domene), a opção de data de pagamento (dois meses ou safra com seus respectivos preços), a quantidade pedida do respectivo produto (cinco), o numero do pedido (0001), um campo referente às observações e informações adicionais que podem ser inseridas pelo usuário e um código do pedido (que será auto-incrementado), estando estes dados disponíveis para consulta e alteração dos dados.

Informações Adicionais:

CORPO

Cenário:

- Ator: Responsável da associação.

- Descrição (Instruções):

1- Ator acessa a página com o objetivo de realizar um pedido;

2- Ator inicia a realização de um pedido selecionando o produtor que efetuará o pedido e a data de realização do pedido;

3- Ator seleciona o produto a ser pedido e os fornecedores que disponibilizam este produto, passando a escolher a data de pagamento, a quantidade de produtos, inserindo o numero do pedido e inserindo alguma observação ou informação adicional sobre o pedido realizado;

4- Sistema deve garantir a correta alternância dos campos e dos botões com a utilização da tecla TAB;

5- Sistema deve garantir a correta inserção dos dados em todos os campos, relacionados ao tipo de dado que o campo espera (número, letra, data);

6- Sistema deve validar as mensagens de correção para o preenchimento incorreto dos campos;

7- Sistema deve validar que o novo cadastro a ser inserido, não cause erro de duplicidade de dados com algum dado do Banco de Dados;

8- Sistema deve validar o número limite de caracteres em todos os campos do *software;*

9- Sistema deve garantir a obrigatoriedade de preenchimento dos campos requeridos;

10- Sistema deve fornecer TITLE de auxilio para o preenchimento de todos os campos do *software*;

11- Sistema deve validar a não inserção de data, quantidade e número do pedido igual a zero;

12- Sistema deve validar a correta inserção de um ano;

13- Sistema deve validar a existência de todos os pop-ups de confirmação para todas as ações necessárias;

14- Ator deverá clicar no botão de ADICIONAR com o objetivo de inserir um novo produto do pedido na tabela que será registrada na base de dados;

15- Sistema deve validar a correta inserção dos dados na tabela, para que não haja perda de qualquer dado;

16- Ator deverá confirmar o cadastro clicando no botão de SALVAR;

17- O sistema deve exibir uma mensagem de confirmação da realização do pedido (Pedido Realizado com Sucesso).

Resultado Esperado: Pedido registrado no banco de dados relacionando o produtor com data de pedido, os produtos do pedido e seus respectivos fornecedores, selecionando a data de pagamento e seus valores, com a quantidade de cada produto e o numero do pedido, isto sem causar conflito com qualquer outro registro que já estiver cadastrado na base de dados, estando os dados disponíveis para realização de consultas e alterações no registro.

APÊNDICE – D

Neste apêndice será apresentado o questionário que foi utilizado para a validação da utilização da técnica de TDD pela equipe de desenvolvedores do software.

QUESTIONÁRIO:

Com relação à utilização do Desenvolvimento Orientado a Testes responda as seguintes questões relacionadas ao conhecimento empírico adquirido:

1. Com o modelo de caso de teste que auxilia na elaboração das telas, como você considera sua compreensão para criação das telas?

   ( ) Melhorou ( ) Não houve alteração ( ) Piorou

   Justifique:_____
_____
_____
_____

2. Com o modelo de casos de teste que auxilia na refatoração do software, como você considera sua compreensão sobre o que deve ser implementado?

   ( ) Melhorou ( ) Não houve alteração ( ) Piorou

   Justifique:_____
_____
_____
_____

3. Utilizando os modelos de criação das telas e de refatoração do código, como você considera a velocidade de desenvolvimento do *software*?

   ( )Melhorou ( ) Não houve alteração ( ) Piorou

Justifique:_____
_____
_____
_____

4. Identifique uma dificuldade ao utilizar a técnica de TDD para o desenvolvimento.

   Justifique:_____
_____
_____
_____
_____

5. Após trabalhar com a técnica de TDD, como você analisa o desenvolvimento de software entre as duas metodologias que você já utilizou? Se pudesse optar, qual você prefere utilizar?

   Justifique:_____
_____
_____
_____
_____