UNILASALLE

CENTRO UNIVERSITÁRIO LA SALLE

ROSANE MARTINS VEBER MOREIRA

AVALIAÇÃO E APLICAÇÃO DE TÉCNICAS DE ESTIMATIVA DE ESFORÇO EM PROJETOS DE TESTES DE SOFTWARE

CANOAS, 2009

ROSANE MARTINS VEBER MOREIRA

AVALIAÇÃO E APLICAÇÃO DE TÉCNICAS DE ESTIMATIVA DE ESFORÇO EM PROJETOS DE TESTES DE SOFTWARE

Trabalho de conclusão apresentado para banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle - Unilasalle, como exigência parcial para a obtenção do grau de Bacharel em Ciência da Computação em 25/06/2009, sob orientação do Prof Me. Roberto Petry.

CANOAS, 2009

TERMO DE APROVAÇÃO

ROSANE MARTINS VEBER MOREIRA

AVALIAÇÃO E APLICAÇÃO DE TÉCNICAS DE ESTIMATIVA DE ESFORÇO EM PROJETOS DE TESTES DE SOFTWARE

Trabalho de conclusão aprovado como requisito parcial para a obtenção do grau de Bacharel em Ciências da Computação do Centro Universitário La Salle - Unilasalle, pela seguinte banca examinadora:

Prof. Marcos Ennes Barreto
Unilasalle

Prof. Rafael Kunst
Unilasalle

Canoas, 25 de junho de 2009.

DEDICATÓRIA

Dedico este trabalho a meus pais que me ensinaram a ter a determinação necessária para atingir os objetivos na vida e a nunca desistir diante de dificuldades.

A meu esposo Marcos que me deu o apoio necessário durante este percurso e a nossa filhinha Luíza que está quase chegando.

AGRADECIMENTO

Gostaria de agradecer a Deus, que me permitiu chegar até este momento, colocando em minha vida pessoas tão especiais que me ajudaram a crescer e a concluir minha graduação.

Dentre essas pessoas, gostaria de iniciar agradecendo aos meus pais, Alfredo e Cláudia, por toda a confiança e compreensão que sempre depositaram em mim.

Ao meu esposo, amigo e companheiro, Marcos, por todo o carinho, paciência e força demonstrados a cada dia.

Ao meu orientador, professor Roberto Petry, pelo apoio e prontidão, especialmente neste momento final da graduação.

Aos familiares, amigos e colegas que tiveram uma participação fundamental em toda minha graduação, sempre dispostos a me ajudar.

Ao meu amigo Sidney Ferraz pela amizade e apoio durante a conclusão deste projeto.

A todos os professores do curso de Ciência da Computação do Unilasalle - Canoas, responsáveis pela minha formação.

A empresa que gentilmente cedeu os dados necessários para a execução deste projeto, bem como todos os funcionários que ajudaram através de sua prontidão e experiência.

Por fim, a todos os que torcem pelo meu sucesso.

Muito obrigada a todos.

EPÍGRAFE

“Seja simpático com os estudiosos - aqueles estudantes que os demais julgam que são uns idiotas. Existe uma grande possibilidade de vocês um dia virem a trabalhar pra eles.” (Bill Gates).

“Inteligência é a habilidade de evitar fazer o trabalho, e mesmo assim conseguir ter o trabalho realizado.” (Linus Torvalds)

RESUMO

Existe grande necessidade de aperfeiçoar as estimativas de esforço em projetos de testes de *software*, devido à ausência de utilização de técnicas específicas para estas atividades, durante o processo de desenvolvimento de *software*. Esta dificuldade gera, na maioria das vezes, estouro nos prazos de entrega e conseqüentemente elevado custo, além de comprometer a qualidade do produto quando não testado devidamente em um tempo adequado. No entanto, estimar tempo e recursos a serem alocados para testes não é uma tarefa fácil, tanto que especialistas em gerenciamento de projetos têm encontrado dificuldade em escolher e aplicar o método adequado nesta área. Este trabalho tem como objetivo descrever a aplicação das técnicas de estimativa de esforço Análise de pontos de função, Pontos de caso de uso, Wideband delphi e Análise de pontos de teste, em projetos de testes em uma fábrica de sofrware. Após, faz uma comparação dos resultados estimados com os tempos efetivamente consumidos pela equipe para executar os testes e, desta forma, oferecer informações relevantes para analisar os métodos utilizados.

Palavras-chave: Estimativas de esforço. Testes de *Software*.

ABSTRACT

There is a great necessity to improve the effort estimates in software testing projects, due to a lack of use of specific techniques for these activities, during the software process development. This difficulty generates, in most cases, unfulfilled deadline and consequently high cost, besides of compromising the quality of the product when it is not tested correctly in an adequate space of time. However, estimate time and resources to be allocated for tests is not an easy task, so much that specialists on project management have found difficulty to choose and to apply the adequate method in this area. This paper aims to describe the estimate techniques application of Function Points Analysis effort, Use Case Points, Wideband Delphi and Test Points Analysis in projects tests at a software factory. After that, the estimated results will be compared with the time effectively consumed by the staff to develop the tests and, this way, offer relevant information to analyze the methods used.

Key-words: Effort estimate, Software tests

SUMÁRIO

1 INTRODUÇÃO

A globalização da economia tem contribuído para que haja uma maior competitividade no mercado entre as empresas produtoras e prestadoras de serviços de software. Isto fez com que a demanda pela produção de *software* aumentasse intensa e sistematicamente, nos últimos anos, devido à exigência de informações atualizadas quase que instantaneamente, criando uma dependência de seus sistemas. Além disso, o *software* vem evoluindo à medida que novos ambientes, plataformas, metodologias e um número cada vez maior de usuários se envolve com o desenvolvimento de sistemas.

Desta forma, as equipes sofrem pressões para construir sistemas em tempo hábil, considerando que os testes sejam realizados e o produto entre no mercado sem reduzir a qualidade adequada e, desta forma, evitando reações negativas nos consumidores e na imagem da empresa (NAGESWARAN, 2001, p. 2). Assim sendo, a atividade de testes passou a ser uma etapa importante no processo de desenvolvimento de *software* à medida que as empresas passaram a procurar novos caminhos para melhorar a qualidade de seus sistemas. Foi necessária a criação de processos apropriados para esta etapa que possui metodologias próprias e equipes independentes (RIOS; MOREIRA FILHO, 2007, p. 288).

Portanto, a qualidade deve estar presente não só nos produtos produzidos, como também nos processos utilizados para o desenvolvimento e manutenção destes produtos. De acordo com a norma ISO/IEC 9126 um produto de *Software* é definido como uma entidade de *software*

disponível para liberação a um usuário. Enquanto que a qualidade de *software* é conceituada como totalidade das características de um produto de *software* que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas. Em geral, as necessidades explícitas são expressas na definição de requisitos propostos pelo produtor, e as necessidades implícitas são aquelas que não estão expressas nos documentos do produtor, mas que são necessárias para o usuário (ISO/IEC 9126 2001).

Neste contexto, um dos fatores responsáveis pela qualidade no processo de desenvolvimento de *software* é o planejamento, que engloba um conjunto de atividades, como por exemplo: a) estimativas (tempo, custo, riscos); b) determinar o cronograma, fazer o planejamento organizacional, análise e gestão de riscos.

Visto que muitos especialistas em gerenciamento de projetos têm dificuldades em estimar o tempo e os recursos a serem alocados para testes, para ajudar na escolha do método adequado, faz-se necessária a análise de modelos de estimar o esforço para testes. Dentre os métodos mais utilizados para estimar esforço de testes, pode-se citar a opinião de especialistas (empirismo) e o julgamento baseado em históricos de projetos anteriores. Embora estes auxiliem nesta tarefa , não podem ser considerados tão precisos quanto à obtenção de valores reais baseados em cálculos previamente definidos (RIOS; MOREIRA FILHO, 2007, p. 227).

Sendo assim, é importante utilizar técnicas precisas e mais apuradas da estimativa de esforço, para que valores superestimados não elevem os prazos e os custos dos projetos, prejudicando a competitividade das empresas de desenvolvimento de *software*. Ao mesmo tempo, valores subestimados podem gerar cronogramas mal dimensionados e com possibilidades de perdas ou prejuízos financeiros.

Por fim, o objetivo deste trabalho é aplicar as metodologias Análise de Pontos de Função, Pontos de caso de uso, Pontos de teste de *software* e Wideband delphi em projetos de teste de *software*, visando comparar o esforço previsto com o realizado. Desta forma, pode-se iden-

tificar as técnicas que mais se adaptam a estes projetos, com a finalidade de apoiar o gerenciamento dos mesmos.

1.1 Estrutura do trabalho

Após este capítulo introdutório e para uma melhor explanação das ações realizadas para alcançar os objetivos, esse documento está disposto da seguinte maneira:

Capítulo 2 - **Processo de desenvolvimento de *software*:** apresenta resumidamente, os conceitos do processo de desenvolvimento de *software*, e descreve os principais modelos de ciclo de vida de *software*.

Capítulo 3 - **Testes de *Software*:** tem a finalidade de conceituar o teste de *software*, bem como descrever as etapas do processo de testes. Além disso, apresenta os métodos utilizados nesse processo, bem como, os níveis e as classificações existentes.

Capítulo 4 - **Estimativas:** descreve uma breve introdução sobre estimativas e apresenta algumas técnicas de estimar esforço de *software*.

Capítulo 5 - **Estudo de Caso:** este capítulo apresenta o enfoque principal deste trabalho, isto é, a aplicação de técnicas de estimativa de esforço: análise de pontos de teste (APT), wideband delphi e pontos por caso de uso (UCP), além das estimativas obtidas com o método análise de pontos de função (APF). A seguir, os resultados são comparados com o esforço realizado em projetos de teste de *software* na busca de aperfeiçoamento do problema proposto.

Capítulo 6 - **Conclusão e Trabalhos Futuros:** tem como objetivo apresentar as conclusões sobre o estudo realizado, além de relatar as dificuldades encontradas e fazer uma consideração sobre trabalhos futuros.

2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Processo de *software* é um conjunto de atividades e resultados associados que levam à construção de um produto de *software*. Pode envolver o desenvolvimento desde o início ou quando um *software* é desenvolvido mediante a expansão e a modificação de sistemas já existentes (SOMMERVILLE, 2003, p.36,37). Há diversos processos de *software*, porém todos com atividades em comum, como: especificação, projeto e implementação, validação e evolução de *software*. A figura 1, apresenta um modelo de processo de desenvolvimento de *software* e as etapas do processo de teste em pararelo:

A seguir, apresenta-se as etapas dos processos de desenvolvimento de *software* e suas atividades, que são definidas nas diversas propostas de modelos de ciclo de vida de *software*.

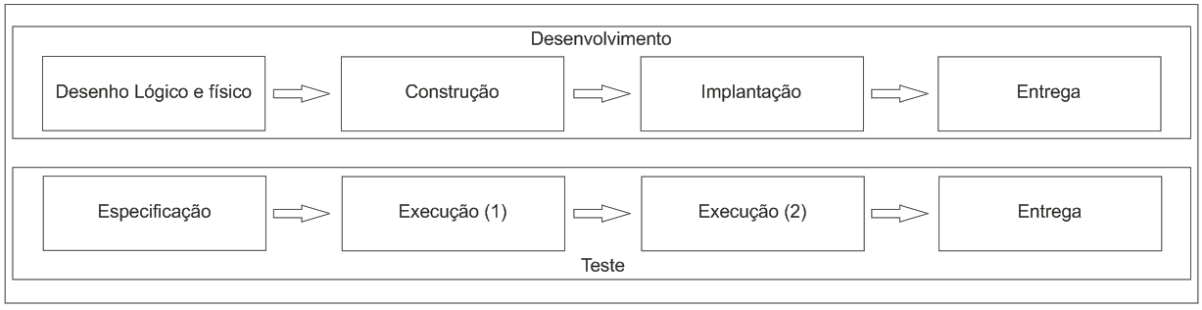

Figura 1 - Modelo de processos de desenvolvimento de software
Fonte: Rios; Moreira Filho, 2007, Adaptada pelo autor deste trabalho

2.1 Modelos de ciclo de vida de software

Modelos de ciclo de vida consistem nas etapas do processo de desenvolvimento de siste-

mas e as atividades a serem realizadas em cada etapa (PETERS; PEDRYCZ, 2001, p. 40-42). O estudo do processo de desenvolvimento provocou o surgimento de várias propostas de ciclo de vida que incluem: o modelo cascata, modelo iterativo e espiral, entre outros.

Segue uma breve descrição destes modelos:

Cascata: O modelo de ciclo de vida cascata foi o primeiro modelo a ser conhecido em engenharia de *software*. Consiste num modelo linear, em que as fases são executadas sistematicamente de forma sequencial, ou seja, a fase seguinte só deve iniciar após a anterior ser concluída. Por exemplo, a análise de requisitos deve ser completada antes que o desenho do sistema possa ser iniciado. Os nomes dados a cada passo variam, assim como a definição exata de cada um deles, mas basicamente o ciclo de vida começa com a análise de requisitos movendo-se em seguida para a fase de projeto, implementação, teste e manutenção do sistema (PRESSMAN, 1995, p. 32,33).

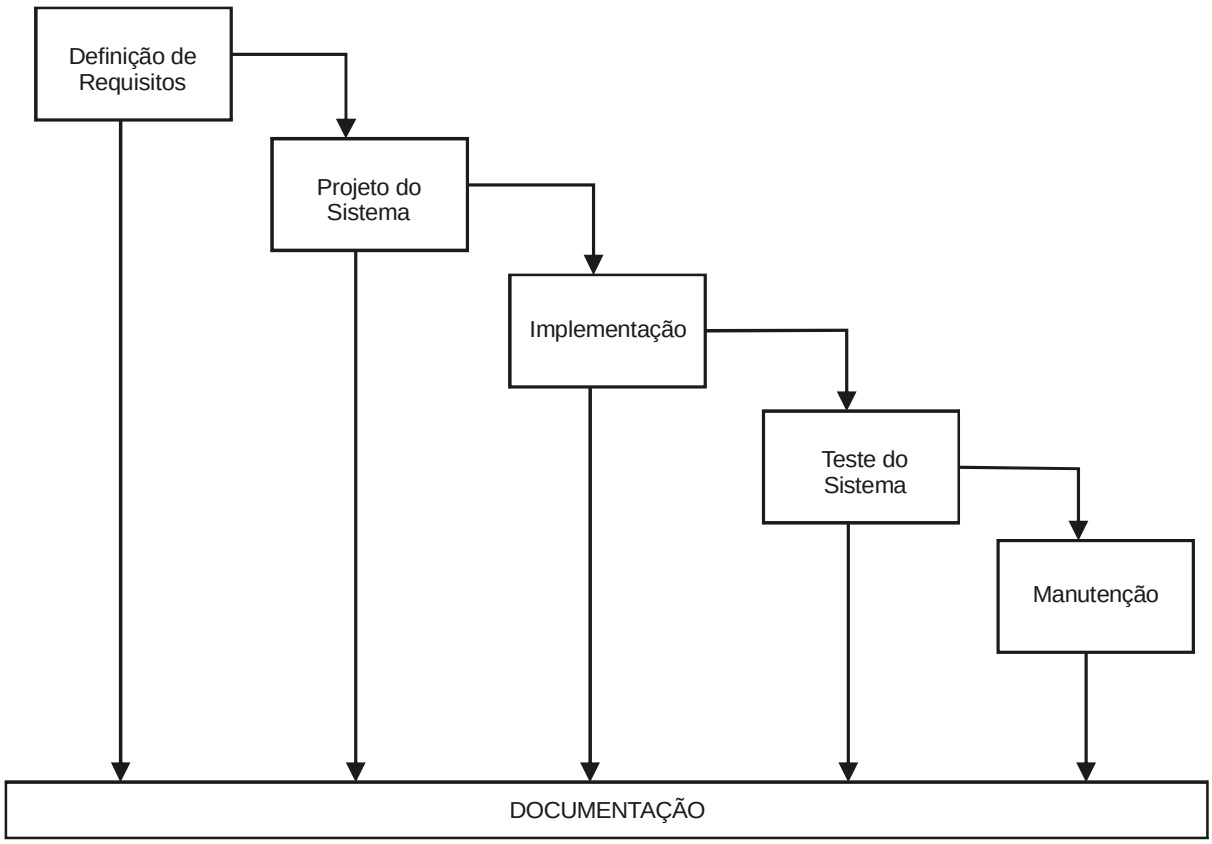

Figura 2 - Modelo cascata
Fonte: www.macoratti.net/proc_sw1.htm

De acordo com a figura 2, seguem as definições das etapas do modelo cascata:

a) Análise e definição de requisitos: São definidos os objetivos, funções e as restrições, com ajuda de clientes e usuários, e servem como uma especificação do sistema, indicando o que deve ser implementado.
b) Projeto do sistema: Envolve a descrição do sistema e do *software* em termos de unidades abstratas e de suas relações, indicando como o *software* deve ser implementado.
c) Implementação: As unidades do *software* devem ser codificadas individualmente.
d) Teste do sistema: As unidades são integradas e testadas.
e) Entrega, operação e manutenção: O sistema é instalado e colocado em operação. A manutenção envolve a correção de erros e evolução do sistema para atender a novos requisitos.

Uma das grandes falhas deste modelo é o fato de os requisitos estarem constantemente mudando, fazendo com que seja dificil voltar atrás para corrigir os erros. Ou seja, o modelo em cascata é apropriado quando se tem um entendimento claro dos requisitos.

Iterativo e Incremental: Um ciclo de vida iterativo se baseia no aumento e refinamento sucessivo de um sistema através de múltiplos ciclos de desenvolvimento da análise, projeto, implementação e de teste, conforme mostra a figura 3. É uma estratégia de planejamento de retrabalho em que o tempo de revisão e melhorias de partes do sistema é pré-definido. É através dos vários ciclos, no desenvolvimento iterativo, que o sistema é refinado e ajustado ou que são adicionados novos requisitos. Cada ciclo trata de um conjunto relativamente pequeno de requisitos.

No Desenvolvimento Incremental várias partes do sistema são desenvolvidas em paralelo, e integradas quando completas. O objetivo do desenvolvimento incremental é adicionar funcionalidades a um sistema durante vários ciclos de liberação de produto. O produto pode ser composto de múltiplos ciclos de desenvolvimento iterativo e cada produto contém mais funcionalidades que o anteriormente gerado.

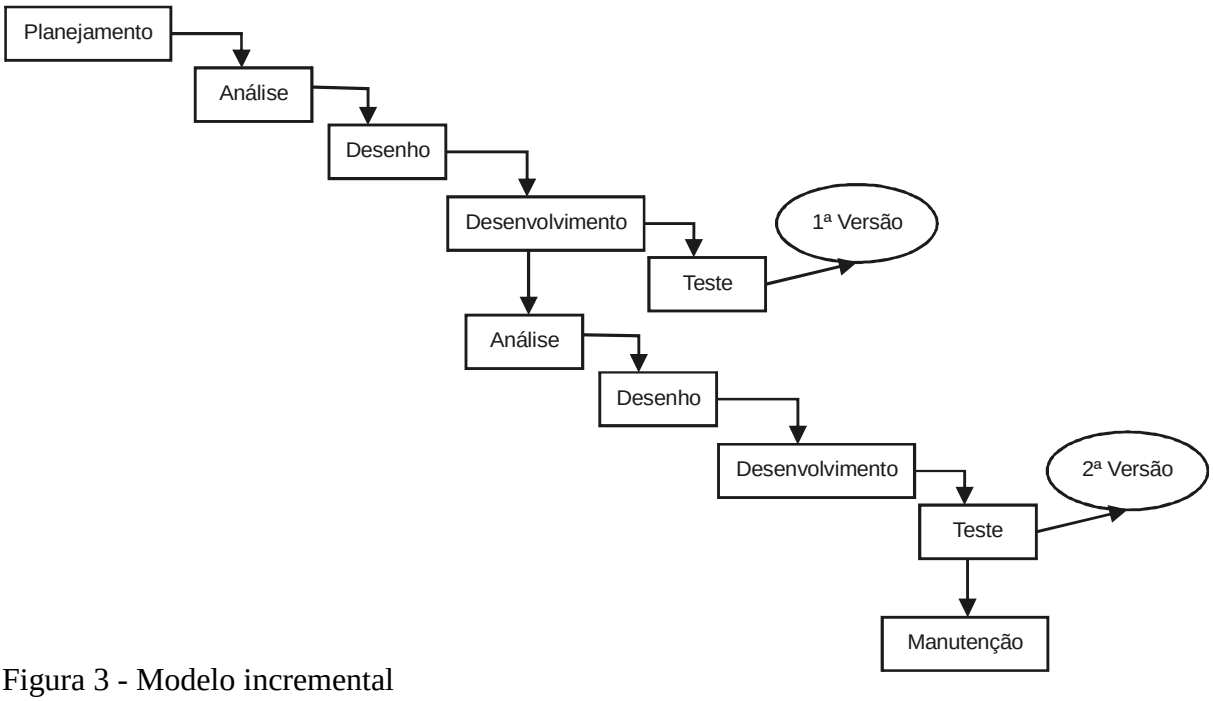

Figura 3 - Modelo incremental
Fonte: www.macoratti.net/proc_sw1.htm

Embora neste modelo seja difícil realizar mudanças de requisitos com o processo em andamento, possui muitas vantagens, conforme descritas a seguir:

a) Padronizar os métodos para cada fase;

b) As etapas são semelhantes as genéricas aplicáveis a todos os paradigmas;

c) É apropriado adotá-lo quando se tem um entendimento claro dos requisitos;

d) Fornece a possibilidade de avaliar os riscos e pontos críticos do projeto mais cedo, além de identificar medidas para os eliminar ou controlar;

e) Possui uma definição de arquitetura que oriente melhor o desenvolvimento;

f) Reduz os riscos envolvendo custos a um único incremento, ou seja, se houver necessidade de a equipe repetir a iteração, a organização perde somente o esforço mal direcionado de uma iteração;

Espiral: O processo é representado como uma espiral, onde cada ciclo do espiral é uma fase do processo. Este modelo fornece uma estrutura de trabalho para a produção de *software* baseada em processo e níveis de risco permitindo a análise dos riscos em várias etapas do desenvolvimento. Não há fases fixas pré-definidas. Elas são definidas de acordo com os objetivos. É

considerado um meta-modelo, pois pode abranger diferentes outros modelos, desde as características do modelo cascata até os vários tipos de protótipos. Possui as etapas de planejamento, análise de riscos, engenharia, construção e liberação e avaliação do cliente (PETERS; PEDRYCZ,2001, p. 40-42).

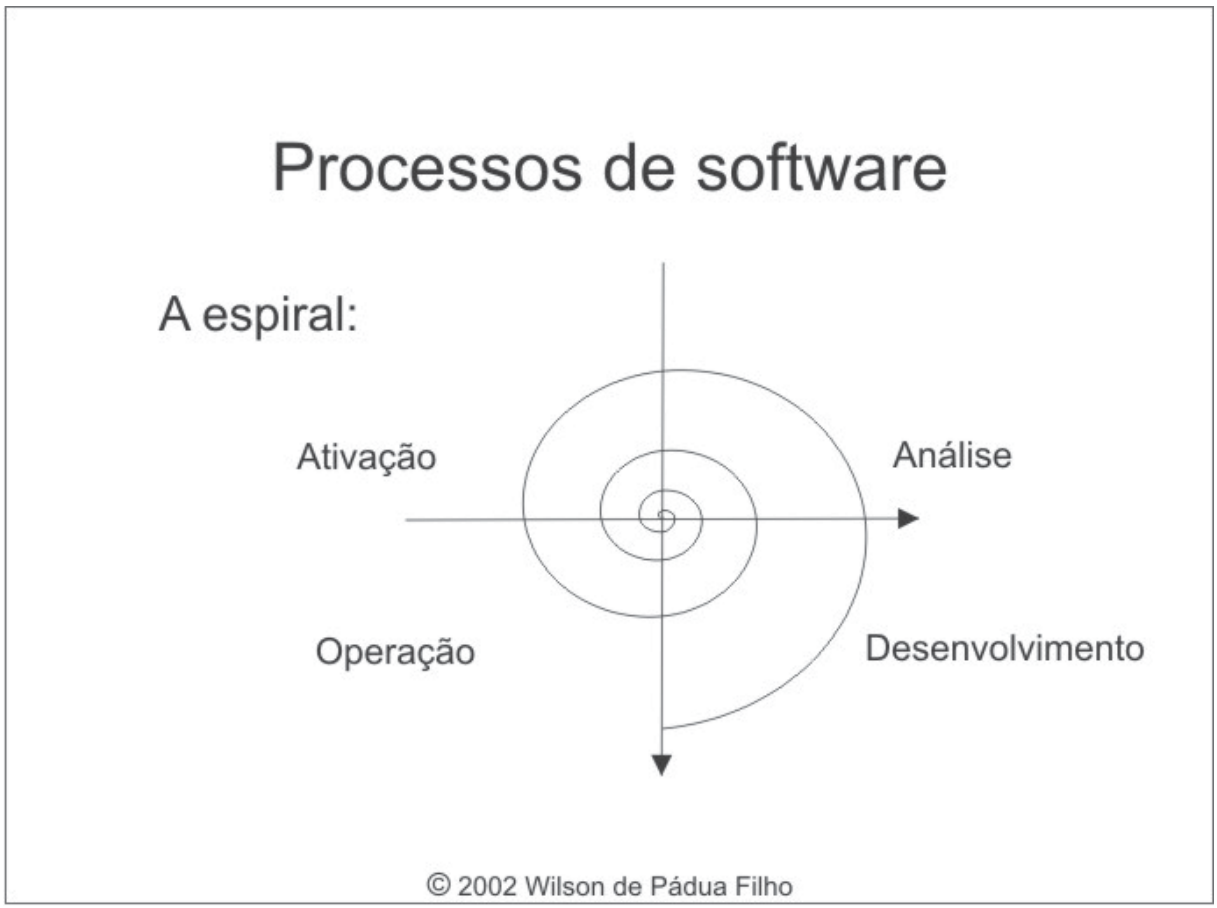

Figura 4 - Modelo espiral
Fonte: www.macoratti.net/proc_sw1.htm

A figura acima indica que este modelo organiza o desenvolvimento como um processo iterativo em que vários conjuntos de quatro fases se sucedem até se obter o sistema final. Um ciclo se inicia com a determinação de objetivos, alternativas e restrições onde ocorre o comprometimento dos envolvidos e o estabelecimento de uma estratégia para alcançar os objetivos. Na segunda tarefa, avaliação de alternativas, identificação e solução de riscos, executa-se uma análise de risco. A seguir, ocorre o desenvolvimento do produto. Na quarta tarefa o produto é avaliado e se prepara

para iniciar um novo ciclo.

O modelo espiral possibilita uma maior integração entre as fases e facilita a depuração e a manutenção do sistema. Além disso, incorpora a análise de riscos e a construção de protótipos a cada ciclo de uma forma iterativa, permitindo o progresso sejam verificados e avaliados constantemente. No entanto, possui as desvantagens de exigir muita experiência para que sejam feitas as avaliações dos riscos e ser um modelo novo e pouco utilizado.

3 TESTE DE SOFTWARE

O objetivo do teste de *software* é a sua execução de forma controlada, a fim de avaliar o seu o comportamento baseado no que foi especificado. Esta execução é considerada um tipo de validação (RIOS; MOREIRA FILHO, 2003, p. 8,9). A atividade de testar o *software* deve ser planejada cuidadosamente, pois um teste de sucesso é aquele que consegue encontrar erros no sistema, embora isso não signifique que o teste elimina completamente a existência destes, e sim que minimiza a chance de existirem no produto final chegando a um nível aceitável pelo cliente.

3.1 O processo de testes

As atividades de teste são uma etapa dentro do processo de desenvolvimento, e por isso, devem basear-se em uma metodologia aderente a esse processo, além de pessoal técnico qualificado, em ambiente e ferramentas adequadas. O processo de teste de *software* segue o conceito "V" de teste (SOMMERVILLE, 2003, p.361), que normalmente é chamado de Verificação e Validação, sendo que a Verificação refere-se ao conjunto de atividades que garante que o *software* implemente corretamente o que foi especificado, enquanto que a Validação garante que o que foi construído está de acordo com as necessidades reais do usuário (PRESSMAN, 1995, p. 836, 837; RIOS; MOREIRA FILHO, 2003, p. 30, 31).

Teste de *software* é uma das atividades de verificação e validação do produto desenvolvi-

do. O ciclo de vida do processo de teste conceito “V” é denominado e composto por quatro etapas seqüenciais (Procedimentos iniciais, Especificação, Execução e Entrega), e duas paralelas (Planejamento e Preparação), baseado na metodologia TMAP (Test Management Approach), conforme indicado na figura a seguir:

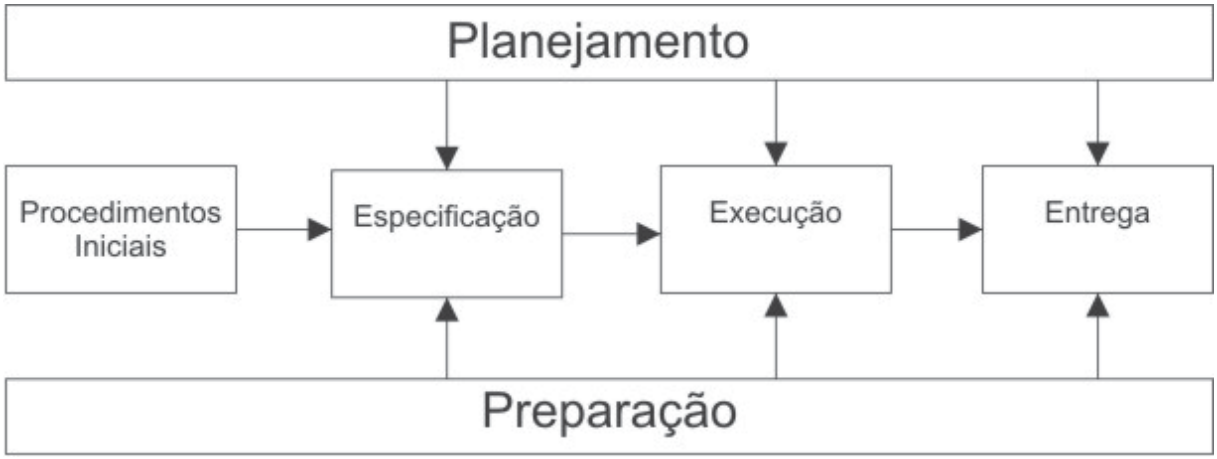

Figura 5 - Fases de um processo de testes
Fonte: Rios; Moreira Filho, 2003, p. 9

Conforme Rios; Moreira Filho (2007, p. 45-47), segue uma descrição das etapas do processo de testes de *software*:

a) **Procedimentos Iniciais:** neste momento, são definidos os objetivos do projeto de testes, a equipe a ser envolvida (desenvolvimento, equipe de testes e usuários), as responsabilidades de cada um, o plano de trabalho, a avaliação dos riscos e os níveis de serviço acordados entre as partes envolvidas, registradas num documento com todas as principais atividades que serão executadas.

b) **Planejamento:** durante o planejamento devem ser elaborados a estratégia e o plano de teste que serão utilizados para minimizar os principais riscos do negócio e encaminhar as próximas etapas. Estas atividades devem ser executadas ao mesmo tempo em que estiverem sendo levantados os requisitos e o planejamento do projeto de desenvolvimento.

c) **Preparação:** nesta etapa ocorre a preparação do ambiente para a execução dos testes, envolvendo equipamentos, equipe (desenvolvedores, testadores e usuários), ferramen-

tas, hardware(máquinas, identificação dos componentes) e *software*(sistema operacional, arquitetura do sistema, browser em aplicação web, etc). Deve ocorrer em paralelo com as outras etapas.

d) **Especificação:** consiste na elaboração e revisão dos casos e dos roteiros de teste, baseados na especificação funcional do sistema. Durante esta fase, os planos de testes devem ser revisados e atualizados, se for o caso.

e) **Execução:** é a fase da execução dos testes especificados nos casos de testes, “scripts” (ferramentas de automação) e dos roteiros de testes, com os correspondentes registros dos resultados obtidos. Para realizar esta fase, deve-se inserir no sistema dados de entradas reais e comparar se os resultados obtidos são semelhantes aos esperados. Normalmente, essas situações de verificação são chamadas “Casos de Testes” e são desenvolvidas na fase de análise dos requisitos do novo produto. Além disso, os testes devem ser executados integralmente, por regressão, ou parcialmente, sempre que surgirem mudanças de versão dos programas em teste e nos ambientes de teste preparados.

f) **Entrega:** consiste na conclusão do processo de testes e entrega do sistema para o ambiente de produção. A documentação deverá ser finalizada com as ocorrências que forem consideradas relevantes para a melhoria do processo. Após, deve ser elaborado um relatório gerencial com as conformidades e não-conformidades encontradas e registradas.

3.2 Classificação dos testes

Existem muitos métodos de testar *software*, que podem ser classificados de acordo com suas características básicas. Dentre estes pode-se descrever os seguintes:

a) **Testes estruturais:** o teste estrutural, também conhecido como teste de Caixa-Branca, deve garantir que a estrutura do *software* seja sólida e que funcione no ambiente que

será instalado, considerando seu contexto técnico. Como o testador tem acesso ao código fonte da aplicação, esse tipo de teste é desenvolvido através da elaboração de casos de teste, de acordo com a estrutura do programa, que cubram todas as possibilidades de um componente de *software*, e da aplicação diretamente ao código. Embora geralmente usados durante a fase de codificação, os testes estruturais podem ser aplicados em todas as fases do ciclo de vida do *software*.

b) **Teste Funcional:** também conhecido como teste caixa preta , se baseia na especificação funcional para derivar casos de testes. Durante os testes funcionais deve ser verificado se todas as funcionalidades especificadas nos documentos de requisitos estão implementadas corretamente. Esta técnica pode ser considerada a base para o projeto de testes, por iniciar durante a fase de especificação de requisitos do sistema e continuar durante os demais níveis do projeto e especificação de interface. Além disso, durante os testes funcionais podem ser encontradas algumas classes de falhas que podem escapar as chamadas técnicas "caixa-branca" (white-box) de teste estrutural (PEZZE; YOUNG, 2008).

c) **Regressão:** essa fase de teste é aplicada quando algum novo componente é inserido no sistema, ou quando alguma modificação nos componentes existentes é realizada e apresentam defeitos ao se juntar com o restante do sistema já testado. Os segmentos já testados devem ser retestados após a implementação de uma mudança em outra parte do *software*. Um conjunto de dados e scripts devem ser mantidos como *baseline* e executado para verificar se as mudanças introduzidas posteriormente não danificaram códigos considerados bons e aceitos. Os resultados esperados do *baseline* devem ser comparados aos resultados após as mudanças. Se houverem discrepâncias, precisam ser resolvidas antes de se iniciar o próximo nível de testes.

d) **Teste de segurança:** o teste de segurança garante que o objetivo do teste e os dados possam ser acessados apenas por determinado(s) usuário(es). Além disso, verifica se todos

os mecanismos de proteção embutidos num sistema o protegerão de acessos indevidos. Para isso, o testador deve tentar acessar o sistema de qualquer maneira, descobrindo, assim, falhas de segurança que possam comprometer o sigilo e a fidelidade das informações.

e) **Teste de estresse:** o sistema é confrontado com situações anormais, onde o testador deve ir além dos limites do sistema. O sistema é executado de forma que exige recursos em quantidade, freqüência ou volume anormais. Os testes devem ser desenhados para que sejam testadas todas as partes do sistema. Por exemplo: determinar se foi alocado espaço em disco e memória suficientes para executar a aplicação; garantir que a capacidade de comunicação seja suficiente para trafegar o volume de trabalho esperado e entrar com mais transações do que as tabelas, as filas, os dispositivos internos de armazenamento conseguem acomodar.

f) **Teste de desempenho:** utilizado para testar o desempenho do *software*, quando executado dentro de um sistema integrado. Às vezes, é combinado com os testes de estresse e pode ser executado durante todo o processo de desenvolvimento.

g) **Usabilidade:** essa técnica visa detectar problemas de interface ou que tornem o *software* pouco intuitivo. Os usuários reais do sistema são incentivados a utilizá-lo em um ambiente monitorado, para verificar a facilidade de uso da interface em questão e se a aplicação é suficientemente amigável para atingir os objetivos do negócio. Esse tipo de teste é subjetivo, pois se baseia na opinião dos usuários, obtidas através de reuniões, entrevistas ou de outras pesquisas.

3.3 Níveis de teste

a) **Testes unitários:** tem o objetivo de garantir que os menores componentes do código criado, atendem às especificações, em termos de características e funcionalidade. É o

estágio mais baixo da escala de testes e são aplicados nos menores componentes de código. Geralmente, estes testes são realizados pelos desenvolvedores.

b) **Testes de integração:** é executado numa combinação de componentes para verirficar se, juntos, eles funcionam corretamente. Os componentes podem ser pedaços de código, módulos, aplicações distintas, clientes e servidores, etc. O objetivo é encontrar falhas provenientes da integração interna das unidades de um sistema. Na maioria das vezes, os tipos de falhas encontradas são de envio e recebimento de dados.

c) **Testes de sistema:** tem a finalidade de executar o sistema sob o ponto de vista do seu usuário final, varrendo as funcionalidades em busca de falhas. Os testes são executados em condições similares àquelas que um usuário utilizará no seu dia-a-dia de manipulação do sistema. De acordo com a política da organização, podem ser utilizadas condições reais de ambiente, interfaces sistemáticas e massas de dados.

d) **Testes de aceitação:** é a fase em que os testes são conduzidos por usuários finais do sistema. Normalmente, são realizados por um grupo restrito de usuários, que simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado. É um teste formal, conduzido para determinar se um sistema satisfaz ou não seus critérios de aceitação e para permitir ao cliente determinar se aceita ou não o sistema. Podem incluir testes funcionais, de configuração, de recuperação de falhas, de segurança e de desempenho.

Após o entendimento de como funcionam os processos de testes de *software*, assim como suas classificações e níveis de execução, é importante conhecer as técnicas existentes para estimar o esforço destas atividades. Desta forma, pode ser possível ajudar na escolha do método mais adequado para estimar o tempo e recursos a serem aocados em projetos de testes.

4 TÉCNICAS DE ESTIMATIVAS

O estabelecimento de estimativas constitui uma das principais atividades do processo de planejamento do projeto, incluindo as atividades de testes. A fim de realizar as etapas descritas acima, é necessário estimar o esforço que será utilizado durante o processo de teste. A realização de estimativas é acompanhada de riscos, considerando alguns fatores como a complexidade e o tamanho do projeto, o grau de estrutura, o escopo mal compreendido e requisitos sujeito a mudanças (PRESSMAN, 2005, p.617-622). Estes fatores afetam a estimativa de todas as etapas do projeto. Portanto, a escolha de técnicas adequadas para estimar o esforço de testes deve ser criteriosa.

A medida de esforço é feita em homens/horas. O projeto deve estimar quantos homens/horas serão necessários para se construir e testar o produto. Diversas técnicas são utilizadas para estimar o esforço nos projetos de *software*. Dentre elas pode-se citar: Análise de Pontos de Função (APF), Análise por pontos de testes (APT), Wideband Delphi e Pontos por Casos de Uso (UCP). Segue a descrição de cada uma das técnicas citadas:

4.1 Análise de pontos de função (APF)

A APF foi desenvolvida em 1979 por Allan Albrecht (KARNER, 1993), na IBM, e posteriormente, em 1986 refinada pelo *International Function Point Users Group* (IFPUG). A partir

da crescente aceitação pelo mercado, surgiram diversas variações da proposta apresentada inicialmente por Allan Albrecht. Com a finalidade de padronizar a técnica, em 1990 foi lançada a primeira versão do CPM – Manual de Práticas de Contagem, e a partir daí este é o padrão reconhecido pela indústria para pontos de função.

A técnica da análise de pontos de função permite uma contagem no início do desenvolvimento sem conhecer detalhes do modelo de dados e posteriormente na fase de projeto é revista. São contados basicamente: as funções relacionadas aos dados utilizados e os arquivos lógicos internos (ALI) e os de interface externa (AIE), as funções transacionais como entrada externa (EE), saída externa (SE) e consulta externa (CE). Após, são calculados os Pontos de Função (PF) não ajustados com base no total de ALI, AIE, EE, SE e CE, determinando o fator de ajuste baseado nas 14 características gerais do *software* (indicadores do grau de dificuldade para construir o *software*) e, finalmente, é aplicada uma fórmula para determinar os PF não ajustados.

A seguir, são descritas as 14 características gerais do *software*, consideradas nesta técnica:

a) **Comunicação de dados**: determinar que protocolos são utilizados pelo aplicativo para o recebimento ou o envio de informações;
b) **Processamento distribuído**: definir os aspectos relacionados com processamento e funções distribuídas;
c) **Desempenho:** Indicar os parâmetros estabelecidos pelo usuário quanto a tempos de resposta.
d) **Utilização de equipamento:** identificar o nível de utilização dos equipamentos necessários à execução do aplicativo, incluindo a carga de trabalho exigida pelo aplicativo quando em produção;
e) **Volume de transações**: verificar qual a capacidade do *software* em relação ao volume de transações esperado;
f) **Entrada de dados on-line:** determinar a quantidade de entrada de dados *on-line.*

g) **Atualização on-line**: definir o modo de atualização dos arquivos lógicos internos do aplicativo;

h) **Eficiência do usuário final**: verificar como se relaciona a eficiência do aplicativo na interação com o usuário. A pontuação é decidida quanto aos recursos implementados para tornar o aplicativo amigável;

i) **Processamento complexo**: decidir quais os detalhes específicos devem ser considerados para a pontuação como processamento lógico extensivo e processamento matemático extensivo.

j) **Reusabilidade de código:** determinar os aspectos relacionados à reutilização do código do aplicativo.

l) **Facilidade de implantação:** encontrar o grau de dificuldade de implementação do aplicativo. Verificar planos de conversão e de implementação;

m) **Facilidade Operacional:** avaliar os procedimentos operacionais automáticos e mecanismos de iniciação, salvamento e recuperação de dados;

n) **Facilidade de mudança**: definir o grau de flexibilidade do aplicativo com relação a mudanças de requisitos do usuário.

o) **Múltiplos locais**: identificar os aspectos relacionados com a arquitetura do aplicativo e a necessidade de instalação em vários lugares;

A APF é uma das medições de estimativas de tamanho mais sedimentadas no mercado, e proporciona resultados mais precisos à medida que artefatos da fase de análise e projetos são gerados (SBQS 2004).

4.2 Wideband delphi

É um método que faz consultas a especialistas de uma determinada área, em uma linguagem e/ou assunto, para elaborar estimativas utilizando a experiência e o entendimento do projeto

proposto. É composto por 3 pessoas ou mais com as seguintes funções: um gerente, um moderador e um ou mais especialistas.

A função do gerente é definir as tarefas do projeto, e escolher o moderador e os especialistas que estarão envolvidos em estimar. O moderador deve coordenar e planejar as atividades, dirigir as reuniões, receber os resultados de cada especialista, preparar um resumo das estimativas e posteriormente apresentar e discutir os resultados com os especialistas. Ao final, apresenta a estimativa mais adequada. Ao especialista cabe fazer a estimativa de cada uma das tarefas e participar das reuniões (BOEHM, 1981).

Esta técnica é composta das seguintes etapas:

a) **Planejamento:** o gerente de projeto deverá escolher o moderador e os especialistas que irão participar da realização da estimativa. Definirá as tarefas que serão estimadas.
b) **Reunião inicial:** todos os envolvidos na estimativa deverão estar presentes. Será passada a especificação do problema, as restrições do projeto, a lista inicial das tarefas. As tarefas podem ser alteradas. Tem duração de aproximadamente uma hora.
c) **Preparação individual:** individualmente cada especialista deverá estimar as tarefas da lista definida na reunião inicial.
d) **Reunião de estimativas:** depois que cada especialista enviar sua estimativa para o moderador será feita uma reunião na qual o moderador apresenta os resultados individuais anonimamente. Conforme recomendado nesta metodologia, as rodadas continuam até as estimativas convergirem ou até no máximo quatro rodadas. Por fim, chega-se à estimativa mais adequada.

O processo Delphi tem como meta compartilhar diferentes visões dos especialistas. Em projetos grandes, sugere-se a aplicação do processo de forma simultânea aos diferentes componentes do produto, ao final, combinam-se as estimativas e elege-se uma estimativa final.

4.3 Análise de pontos de teste (APT)

Esta unidade de mensuração da atividade de teste foi desenvolvida por Martin Pol, Ruud Tennissen e Erik Van Veenendaal, e toma como premissa básica o tamanho do sistema em Pontos de Função considerando todas as suas funcionalidades como ponto de partida. O total de pontos de testes é calculado pela soma dos pontos de teste dinâmicos e estáticos. Os pontos de teste dinâmicos se baseiam nas funções do sistema, tratadas pela análise de pontos de função. Já os pontos de teste estáticos consideram o sistema como um todo e tomam como base os critérios de qualidade para a avaliação de características como funcionalidade, performance, segurança e aderência. Esta avaliação é feita através de *checklists*. Caso a equipe não adote processos de revisão de documentação e de códigos através de *checklist*, os testes estáticos devem ser descartados e terá valor nulo. A Análise de pontos de testes considera outros fatores que podem afetar as atividades de teste: o grau de complexidade dos testes, o nível de qualidade que se pretende alcançar, o grau de envolvimento dos usuários com os testes, as interfaces que as funções em teste têm com os arquivos. Além disso, leva em consideração, o ciclo de reincidência de defeitos no sistema, o nível de cobertura esperado, a experiência e a produtividade da equipe de testes (medido através de indicadores históricos), o grau de automação, a qualidade do ambiente de testes e a qualidade da documentação do sistema e dos requisitos (RIOS; MOREIRA FILHO, 2007, P. 228-246).

No entanto, esta medição de projetos de teste é recomendada quando a medição do tamanho do sistema é feita através da técnica Análise de Pontos de Função.

4.3.1 Total dos pontos de teste (PT)

O número total de pontos de teste (PT) é dado através do somatório dos pontos de teste dinâmicos e pontos de teste estáticos, considerando-se o tamanho do sistema em pontos de fun-

ção e uma constante igual a 500, representando o tamanho mínimo em pontos de função. Seguem os detalhes sobre como encontrar cada um destes subtotais.

$$PT = \sum PTDf + (PF \times PTE) / 500$$

Onde: ∑PTDf = Soma dos pontos de teste de todas as funções, PF = tamanho do sistema todo em pontos de função e PTE = total dos pontos de teste estáticos.

4.3.1.1 Pontos de teste dinâmico (PTD)

Estes pontos de teste são calculados baseando-se nas funções do sistema, através da análise em pontos de função. Cada funcionalidade medida tem outra medida para fins de medição dos testes.

As funções medidas em pontos de função e tratadas em pontos de teste, são: Entradas Externas (EE) ou External Inputs, Saídas Externas (SE) ou External Outputs, Consultas Externas (CE) ou External Inquiries.

Além disso, os pontos de teste dinâmico se baseiam nos seguintes elementos: funções dependentes (Fdf) e qualidade dos requisitos relacionados às características de qualidade a serem testadas dinamicamente (QRd).

A partir destes elementos, utiliza-se a seguinte fórmula de cálculo para o ponto de teste dinâmico:

$$PTDf = PFf \times FDf \times QRd$$

Onde: PFf é o número de pontos da função testada, FDf é o total das funções dependentes e QRd é o total das características explícitas e implícitas.

4.3.1.1.1 Fatores das funções dependentes (FDf)

Importância do usuário (Ue): Este fator serve para medir o grau de importância que o

usuário dá a função a ser testada e pode ser coletado através de entrevistas com os usuários do sistema ou com os desenvolvedores.

Quadro 1- Importância do usuário

Peso	Classificação
3	Baixa
6	Normal
12	Alta

Fonte: RIOS; MOREIRA FILHO, 2007, p. 233

Intensidade de uso (Uy): O usuário deverá estabelecer a intensidade de uso da função. Para isso deve ser levantado o nível de utilização da função durante um intervalo de tempo. Existem funções que são muito usadas e outras cuja utilização é eventual.

Quadro 2 - Intensidade de uso

Peso	Classificação
2	Baixa
4	Normal
8	Alta

Fonte: RIOS; MOREIRA FILHO, 2007, p. 233

Interface (I): O fator de interface mede o nível de inter-relacionamento entre os arquivos e as funções que estão sendo medidas. É necessário considerar quantos arquivos são afetados pela função medida e o número de funções que afetam um arquivo específico. Se a função não modificar nenhum arquivo deve ser dada para ela uma interface baixa.

O quadro a seguir mostra como definir o nível de inter-relacionamento descrito:

Quadro 3 - Nível de inter-relacionamento de interface

Arquivos	Funções		
	1	2 – 5	>5
1	Baixa	Baixa	Baixa
2 – 5	Baixa	Normal	Alta
>5	Normal	Alta	Alta

Fonte: RIOS; MOREIRA FILHO, 2007, p. 234

Após a definição acima, deve ser verificada a sua classificação no quadro que segue:

Quadro 4 - Classififcação de interface

Peso	Classificação
2	Baixa
4	Normal
8	Alta

Fonte: RIOS; MOREIRA FILHO, 2007, p. 234

Complexidade (C): O grau de complexidade da função é dado pelo algoritmo da parte do programa que executa a função, que é medido através da quantidade de comandos de condição existentes. Na ausência destas informações, é razoável utilizar o nível normal.

Para o caso de processos de automação de testes, é possível definir um método de captura automática destas informações, quando se tiver uma medida mais precisa.

O quadro a seguir, indica a classificação para o grau de complexidade definido pelas funcionalidades medidas:

Quadro 5 - Complexidade

Peso	Classificação
3	Baixa - até 5 condições
6	Normal - entre 6-11
12	Alta - mais 11

Fonte: RIOS; MOREIRA FILHO, 2007, p. 235

Uniformidade (U): Mede o nível de reutilização do material de teste. Esta medição pode variar entre 0,6 e 1,0, sendo 1,0 a não utilização e 0,6 a completa utilização do material.

Fórmula de cálculo – funções dependentes:

$$FDf = ((Ue + Uy + I + C)/20) \times U$$

Onde: Ue = Importância do usuário, Uy = Intensidade de uso, I = Interface, C=Complexidade e U=Uniformidade

Funções padronizadas: As funções de mensagens de defeito, de ajuda e de estrutura de menu possuem uma fórmula própria de contagem, conforme indicado no quadro seguinte:

Quadro 6 - Funções padronizadas

Funções	Pontos de função	Ue	Uy	I	C	U	Df
Mensagens de defeito	4	6	8	4	3	1	1,05
Telas de ajuda	4	6	8	4	3	1	1,05
Menus	4	6	8	4	3	1	1,05

Fonte: RIOS; MOREIRA FILHO, 2007, p. 236

4.3.1.1.2 Características de qualidade dinâmica (QRd)

Características explícitas: São consideradas medidas explícitas e implícitas para medir a qualidade dos requisitos do sistema, através da fórmula:

$$QRd = CE + CI$$

Temos as seguintes características explícitas com os respectivos pesos: funcionalidade (F), peso 0,75; performance (P), peso 0,10; segurança (S), peso 0,05; aderência e efetividade (A) peso, 0,10.

As características de aderência e efetividade se referem aos testes de alto nível (sistema e aceitação), por isso deve ser avaliado se a tela de uma função está incluída nesta característica.

Para cada característica explícita temos os valores:

Quadro 7 - Valores para características explícitas

0	A qualidade dos requisitos não é importante para o resultado dos testes
3	A qualidade dos requisitos não é importante , mas precisa ser considerada para o resultado dos testes
4	A qualidade dos requisitos tem importância média
5	A qualidade dos requisitos é muito importante
6	A qualidade dos requisitos é extremamente importante

Fonte: RIOS; MOREIRA FILHO, 2007, p. 237

A partir dos valores do quadro acima calculamos cada característica utilizando o seguinte procedimento: F = (Valor atribuído (ver tabela) x 0,75)/4.

Seguindo o mesmo procedimento para as características de performance, segurança e aderência, podemos obter resultado das características explícitas através da fórmula:

$$CE = F + P + S + S$$

Características implícitas: São utilizadas para futuras medições quanto à qualidade dos testes. Estes indicadores devem servir de medida padrão para comparar com o projeto em andamento. Sempre que houver indicadores para avaliar uma das características explícitas (funciona-

lidade, performance, segurança e aderência), pode existir uma característica implícita associada, sendo admitido 1 para cada característica explícita e mantendo-se o limite de 4.

Para calcular as características implícitas, utiliza-se a seguinte fórmula:

$$CI = n \times 0{,}02 \text{ (onde n varia entre 0 e 4)}$$

Sendo assim, QRd será a soma dos fatores explícitos, adicionados de cada uma das características implícitas utilizadas, no limite de 4.

$$QRd = CE + CI$$

4.3.1.2 Pontos de testes estáticos (PTE)

Consideram o sistema como um todo e somente devem ser contados se a equipe de teste adotar processos de revisão de documentação e de códigos através de *checklists*, caso contrário terá valor nulo.

Para calcular os pontos de testes estáticos tomamos como base os critérios de avaliação das características explícitas (funcionalidade, performance, segurança e aderência), utilizando um *checklist* para cada característica, conforme a seguinte fórmula:

$$PTE = 16 \times n$$

Onde, 16 é a quantidade de pontos de teste adicionada para cada *checklist* e n representa o número de *checklists* utilizados para avaliar as características de qualidade do sistema, sendo menor ou igual a 4.

4.3.2 Estimativa das horas de teste primárias (HTP)

O valor da estimativa do número de horas de testes primárias, necessita da avaliação da qualificação da equipe e do ambiente de teste. Para este cálculo, deve ser aplicada a seguinte fórmula:

$$HTP = PT \times QET \times AT$$

4.3.2.1 Qualificação da equipe de teste (QET)

Para determinar o valor da qualificação da equipe, devem ser consideradas a experiência e a qualificação, baseando-se em uma base histórica. Este valor deve variar entre 0,7 a 2,0, considerando que quanto melhor e mais qualificada for a equipe, menor será QET.

4.3.2.2 Ambiente de teste (AT) (Fatores ambientais)

Existem fatores que classificam um ambiente de teste, como ferramentas, precedência, documentação, ambiente e testware, conforme demonstrado nos quadros a seguir:

Quadro 8 - Ferramentas de teste

1	Existe uma ferramenta de automação para as fases de especificação e execução dos testes
2	Existe uma ferramenta de automação para as fases de especificação ou para a fase de execução.
4	Não existe ferramenta de automação de teste.

Fonte: RIOS; MOREIRA FILHO, 2007, p. 243

Quadro 9 - Teste de procedência

2	Existe um plano para o teste precedente e a equipe está familiarizada com ele, assim como com os respectivos casos de teste e resultados.
4	Existe um plano para o teste precedente.
8	Não existe um plano para o teste precedente.

Quadro 10 - Documentação de teste

3	Durante o desenvolvimento do sistema são usados padrões de documentação e templates. Acontecem revisões periódicas da documentação.
6	Durante o desenvolvimento do sistema são usados padrões de documentação e templates. Seu uso não é verificado de maneira formal.
12	A documentação não segue nenhum padrão nem templates são usados.

Fonte: RIOS; MOREIRA FILHO, 2007, p. 244

Fonte: RIOS; MOREIRA FILHO, 2007, p. 244

Quadro 11 - Ambiente de desenvolvimnto

2	O sistema foi desenvolvido usando uma linguagem de 4ª geração integrada a um sistema de gerência de banco de dados.
4	O sistema foi desenvolvido usando uma combinação de linguagens de 3ª e 4ª geração.
8	O sistema foi desenvolvido em linguagem de 3ª geração.
1	O ambiente de teste já foi usado inúmeras vezes.
2	O ambiente de teste é similar ao que já vinha sendo usado anteriormente.
4	O ambiente de teste é completamente novo e experimental.

Fonte: RIOS; MOREIRA FILHO, 2007, p. 244

Quadro 12 - Testware

1	Existem materiais de testes, tais como base de dados, tabelas, casos de testes e outros, que poderá ser reutilizado.
2	Existem apenas tabelas e bases de dados disponíveis para reutilização.
4	Não existe testware disponível.

Fonte: RIOS; MOREIRA FILHO, 2007, p. 244

Fórmula de cálculo:

AT = Soma de todos os fatores / 21

4.3.3 Número total de horas de teste (THT)

A fim de calcular o total de horas de teste, é necessário incluir atividades de planejamento e controle utilizando o Índice de Planejamento e Controle (IPC), que é determinado através do valor percentual da soma de dois fatores: Tamanho da equipe (TE) e Ferramentas de gerência (FG), conforme as fórmulas a seguir:

IPC = 1 + (TE + FG)

Onde: Tamanho da equipe (TE)

Quadro 13 - Tamanho da equipe

Fator	Quantidade de técnicos
0,03	Entre 3 e 4 técnicos (inclusive)
0,04	Entre 5 e 10 técnicos (inclusive)
0,08	Mais de 10 técnicos

Fonte: RIOS; MOREIRA FILHO, 2007, p. 246

Quadro 14 - Ferramentas de gerência

Fator	Disponibilidade de Ferramentas
0,02	Existem ferramentas de registro de tempo e de gerência de defeitos, além de ferramentas de gerência de configuração
0,04	Apenas uma das ferramentas citadas acima está disponível
0,08	Não existem ferramentas disponíveis

Fonte: RIOS; MOREIRA FILHO, 2007, p. 246

A partir deste valor temos o número total de horas de teste, aplicando o cálculo:

THT = HTP X IPC

As atividades de planejamento e controle (IPC) são calculadas em separado, e posteriormente

devem ter seu índice adicionado ao total de horas primárias (HTP) para estimar o esforço total.

4.3.4 Distribuição entre as fases de teste

Embora possam ser utilizados valores a partir de dados históricos coletados, os autores desta metodologia sugerem os seguintes percentuais para a distribuição do esforço entre as fases de teste: Preparação 10%, Especificação 40%, Execução 45%, Transição 5%.

4.4 Pontos por casos de uso (UCP)

A técnica Pontos por Casos de Uso (UCP) foi proposta em 1993 por Gustav Karner, da Objectory (hoje Rational *Software*), com o propósito de estimar o tamanho do sistema ainda na fase de levantamento de casos de uso, utilizando-se dos próprios documentos gerados nesta fase de análise como subsídio para o cálculo dimensional. Ela baseia-se em dois métodos bastante utilizados - o mecanismo de Pontos de Função e uma metodologia conhecida como MARKII, que é uma adaptação da técnica Pontos de função, muito utilizada na Inglaterra.

A metodologia Pontos por Caso de Uso foi apresentada para o mercado como uma solução simples para as estimativas de tamanho, especialmente para os que não conheciam as práticas de Contagem de Pontos por Função ou as consideravam complexas para serem aplicadas. Esta técnica explora o modelo e a descrição do caso de uso, substitui algumas características técnicas propostas pela técnica Análise de Pontos de Função, cria os fatores ambientais e propõe uma estimativa de produtividade. No entanto, tem como desvantagem que só pode ser utilizada por empresas que adotem os casos de uso como forma de expressão dos requisitos. Alguns preferem utilizar esta técnica por ser uma solução simples para as estimativas de tamanho.

A diferença da métrica Pontos por Casos de Uso é a forma de lançar uma estimativa, que

consiste nos seguintes passos:

4.4.1 Classificar o peso dos atores

Os atores envolvidos, são classificados em cada caso de uso, a fim de obter um somatório de pontos não-ajustados. Esta classificação está indicada no quadro 15: o peso total dos atores do sistema (Unadjusted Actor Weight, ou UAW) é calculado pela soma dos produtos do número de atores de cada tipo pelo respectivo peso.

Quadro 15 - Peso de atores

Tipo de ator	Peso	Descrição
Ator Simples	1	Outro sistema acessado através de uma API de programação.
Ator Médio	2	Outro sistema interagindo através de um protocole de comunicação, como TCP/IP ou FTP.
Ator Complexo	3	Um usuário interagindo através de uma interface gráfica (Stand-Alone ou Web)

Fonte: NAGESWARAN, 2001, p. 3, adaptado pela autora

4.4.2 Classificar o peso dos casos de uso

Para realizar o cálculo inicial do peso bruto dos casos de uso (Unadjusted Use Case Weight, ou UUCW), divide-se os casos de uso em três níveis de complexidade, baseando-se no número de transações envolvidas no seu processamento. As transações consistem em uma série de processos que devem ser realizados em conjunto, ou cancelados em sua totalidade, caso não seja possível concluir o processamento. O quadro abaixo mostra o peso para cada um dos tipos de Casos de Uso classificados.

Quadro 16 - Peso dos casos por número de transações

Tipo de Casos de Uso	Número de Transações	Peso
Simples	Até 3	1
Médio	4 a 7	2
Complexo	7 ou mais	3

Fonte: NAGESWARAN, 2001, p. 3, adaptado pela autora

O peso dos casos de uso é calculado através da soma dos produtos da quantidade de casos de uso classificados em cada tipo pelo nominal do tipo em questão.

Outra maneira de se calcular o peso dos casos de uso do sistema é considerar o número de classes envolvidas no processo, conforme indicado no quadro seguinte.

Quadro 17 - Peso dos casos por número de entidades

Tipo de Casos de Uso	Número de Entidades	Peso
Simples	5 ou menos	1
Médio	5 a 10	2
Complexo	mais de 10	3

Fonte: NAGESWARAN, 2001, p. 4, adaptado pela autora

Neste caso, o cálculo é feito da mesma forma que na abordagem anterior, e pode ser aplicado quando já for possível antever as entidades envolvidas em um dado processo.

O peso total não ajustado(Unadjusted Use Case Points) é dado pelo somatório entre os pesos de atores e casos de uso:

$$UUCP = UAW + UUCW$$

4.4.3 Calcular os fatores de ajustes

Este método constitui-se de duas partes – um cálculo de fatores técnicos, cobrindo uma série de requisitos funcionais do sistema; e um cálculo de fatores de ambiente, requisitos não-funcionais associados ao processo de desenvolvimento – como experiência da equipe, motivação e estabilidade do projeto. Estes fatores devem ser aplicados ao peso total não ajustado (UUCP), encontrado anteriormente.

Em ambos os modificadores atribuem-se para cada requisito listado em suas tabelas, um valor que determina a influência entre 0 e 5, sendo que o valor 0 indica nenhuma influência, 3 indica influência moderada e 5 indica forte influência.

4.4.3.1 Calcular os fatores técnicos

Para encontrar o valor de complexidade técnica do sistema (TCF), considera-se os pesos dos fatores descritos no quadro 18.

Quadro 18 - Peso dos fatores técnicos

Fator	Requisição	Peso
T1	Sistema Distribuído	2
T2	Tempo de Resposta	2
T3	Eficiência	1
T4	Processamento Complexo	1
T5	Código reusável	1
T6	Facilidade de instalação	0.5
T7	Facilidade de Uso	0.5
T8	Portabilidade	2
T9	Facilidade de Mudança	1
T10	Concorrência	1
T11	Recurso de segurança	1
T12	Acessível por terceiros	1
T13	Requer treinamento especial	1

Fonte: NAGESWARAN, 2001, p. 5, adaptado pela autora

O cálculo do fator de complexidade técnica do sistema (TCF) é feito pela seguinte fórmula:

$$TCF = 0.6 + (0.01 \times Tfactor)$$

O valor Tfactor é obtido pelo somatório dos níveis de influência atribuídos a cada fator (T1 a T13) multiplicado pelo seu peso correspondente.

4.4.3.2 Calcular os fatores ambientais

Os fatores ambientais previstos para esta metodologia e seus pesos associados são exibidos no quadro abaixo:

Quadro 19 - Peso dos fatores ambientais

Fator	Requisito	Peso
E1	Familiaridade com o RUP ou outros processos formais	1.5
E2	Experiências com Aplicação em desenvolvimento	0.5
E3	Experiência em Orientação a Objetos	1
E4	Presença de analista experiente	0.5
E5	Motivação	1
E6	Requisitos estáveis	2
E7	Desenvolvimento em meio-expediente	-1
E8	Linguagem de programação difícil	2

Fonte: NAGESWARAN, 2001, p. 6, adaptado pela autora

Estes fatores de ajustes correspondem ao nível de disponibilidade do recurso no decorrerdo projeto. Assim, determinar que um fator tem nível de influência alta (atribuir a ele o valor 5), significa que este fator está presente no projeto como um todo e influencia seu desenvolvimento. Por outro lado, a atribuição de um valor de influência zero (nenhuma influência) a um fator, indica que o mesmo não está presente no processo de desenvolvimento.

O cálculo do fator ambiental EF é realizado da mesma forma que o fator TCF. Os valores 1,4 e 0,03 também são pré-definidos pela técnica juntamente com os pesos da primeira coluna, conforme a seguinte fórmula:

$$EF = 1.4 + (-0.03 \times Efactor)$$

Encontra-se o valor de Efactor pela soma dos produtos entre o peso de cada fator (E1 a E8) e seu grau de influência atribuído, como no cálculo da variável Tfactor, abordada anteriormente.

4.4.4 Calcular o porte do sistema

Para calcular o valor total do sistema em Use Case Points (UCP) ajustados, utilizamos a seguinte fórmula:

$$UCP = UUP \times TCF \times EF$$

O cálculo do UCP final, que corresponde ao porte do sistema sugerido pela técnica, é resultado da multiplicação dos pesos dos atores e dos casos de uso, pelos fatores de ajustes técnicos e ambientais.

Segundo Karner, (1993), podemos estimar o tempo necessário para o desenvolvimento do projeto calculando-se uma média de 20 horas de trabalho por Ponto de Caso de Uso (UCP), sendo que Freire e Cândido (FREIRE, 2003; CÂNDIDO, 2005) mencionam que experiências práticas no uso desta técnica demonstram uma variação entre 15 e 30 horas por ponto.

Com o resultado desta medição e sabendo-se a produtividade média da organização para produzir um UCP, pode-se então estimar o esforço total para o projeto.

5. ESTUDO DE CASO

Após analisar as técnicas de estimativas Wideband Delphi, Análise de Pontos de Teste e Pontos por Caso de Uso, descritas neste trabalho, foi realizada a aplicação destas em cinco projetos disponibilizados por uma fábrica de *software*.

A metodologia utilizada nesta pesquisa pode ser classificada quanto ao seu objetivo como sendo exploratória, pois tem a finalidade de identificar o problema, levantar teorias e práticas para serem aplicadas e modificar as práticas existentes, fornecendo as inovações tecnológicas necessárias. Quanto ao procedimento, caracteriza-se como um estudo de caso em uma organização de *software*, e este estudo é de natureza aplicada, por gerar conhecimento para aplicações práticas de problemas específicos.

A seguir, serão descritos os processos de desenvolvimento e testes dos projetos na empresa em estudo, as estimativas atuais, a documentação utilizada, os recursos disponíveis, a arquitetura do sistema em construção, bem como os resultados obtidos após a aplicação das técnicas abordadas neste trabalho.

5.1 A empresa, processos e atividades

Este estudo prático foi realizado em uma empresa de desenvolvimento de *software*, fundada há mais de 15 anos, de porte médio, que busca obter o nível 2 de certificação do CMMI.

Foram selecionados cinco projetos de uma mesma fábrica de *software*, para realizar as estimativas de esforço de teste, através da aplicação das técnicas wideband delphi, análise de pontos de testes e pontos por caso de uso, a fim de estimar com mais precisão o esforço destas atividades. A partir das estimativas encontradas com estas metodologias, calculadas durante este trabalho, juntamente com os resultados fornecidos pela empresa baseados em Pontos de Função, fez-se uma comparação com o esforço realizado para as atividades de testes dos projetos envolvidos.

5.1.1 Estimativas atuais

Atualmente, a empresa utiliza a técnica MARK II, uma variação de Pontos de função, para estimar o tamanho do sistema desta fábrica. A diferença entre estas duas técnicas é que na primeira além das funções de dados (Arquivos Lógicos Internos e Arquivos de Interface Externa) e funções de transação (Entradas Externas, Saídas Externas e Consultas Externas), foi embutido o item de análise de algoritmos. Propõe resolver as dificuldades que a técnica de Análise de Pontos de Função apresenta quando conta sistemas em tempo real de controle de processos e outros que apresentam alta complexidade algorítmica. Além disso, o método também reduz os pesos empíricos das funções de dados e de transação.

A partir desta medida inicial, o tempo para cada fase é dividido em percentuais, considerando a experiência de especialistas e uma base histórica para o processo de desenvolvimento. Esta técnica tem sido eficaz para estimar o esforço de desenvolvimento de seus projetos, porém não atinge os objetivos no caso de esforço de execução de testes de *software*.

5.1.2 Processo de desenvolvimento

O processo inicia, assim que o cliente disponibiliza os documentos de especificações funcionais e técnicas para a fábrica. Uma equipe de analistas de sistemas faz uma revisão

nos documentos fornecidos, verifica a qualidade e consistência da documentação, e solicita as alterações aos analistas que criaram o documento, caso seja necessário. Após, encaminham para os projetistas efetuarem a especificação técnica que deverá ser usada pela equipe de desenvolvimento. Em seguida, é iniciada a construção do *software*.

Ao finalizar esta etapa, antes de liberar a versão do sistema construído para a equipe de testes, os desenvolvedores executam um *checklist* para verificar se o sistema está atendendo aos padrões mínimos solicitados.

A fábrica em estudo, adota o modelo do ciclo de vida iterativo e incremental, para o desenvolvimento de seus projetos.

5.1.3 Processo de testes

À medida que os casos de usos construídos são liberados pela equipe de desenvolvimento, iniciam as atividades de execução de testes.

São recomendados para estes projetos, no mínimo três ciclos. No primeiro, são executados testes de validação de padrões de interface, conforme *checklist* definido pelo cliente (ver ANEXO A), e os casos de teste (ver ANEXO B) para cada caso de uso. No ciclo seguinte, após as correções dos defeitos, são re-testadas as não-conformidades registradas no primeiro ciclo. Somente depois de corrigidos e validados todos os erros encontrados nos dois primeiros ciclos, é que se inicia o terceiro ciclo, onde são realizados novamente todos os testes funcionais.

Os testes funcionais, são realizados manualmente, baseados nas especificações do cliente. Não são executados testes automatizados nestes projetos.

Após o encerramento das atividades de teste, ou seja, quando não existirem defeitos com severidade classificada como alta ou média, o pacote será liberado para o cliente executar os testes de aceitação do sistema.

5.1.4 Documentação

A especificação dos requisitos do sistema é documentada através de casos de usos com atores e ações bem definidos (ver ANEXO C). Os requisitos são considerados estáveis pela equipe de analistas. E, quando ocorrem mudanças após o início do desenvolvimento, é realizada uma análise de impacto para as atividades de desenvolvimento e testes e faz-se uma nova estimativa.

Os casos de testes e o *checklist* de padrões de interface, são criados e mantidos pela equipe de analistas de testes e padrõesde responsabilidade do cliente.

5.1.5 Recursos

A empresa utiliza a ferramenta Testlink, que tem a finalidade de gerenciar e executar os testes e registrar o resultado dos mesmos. Inicialmente foi utilizada apenas como repositório dos casos de testes, e os resultados da execução dos testes eram registrados em planilhas. Porém, a partir do projeto 5, esta ferramenta passou a ser utilizada como gerenciador de planejamento e execução de testes, permitindo ao gestor do projeto delegar as tarefas a cada testador e acompanhar os seus resultados.

O sistema MANTIS é utilizado para apoiar no registro e gerenciamento dos defeitos encontrados.

A equipe de testes envolvida nos projetos 1, 2, 3 e 4 foram de 3 pessoas, composta por 2 analistas de testes e 1 testador. Enquanto que para o projeto 5 foram utilizados 4 recursos, sendo 1 analista de testes e 3 testadores.

5.1.6 Arquitetura

O sistema é cliente-servidor é e dividido em múltiplas camadas, provendo uma clara divi-

são entre as funcionalidades de cada uma delas. A figura a seguir identifica estas camadas.

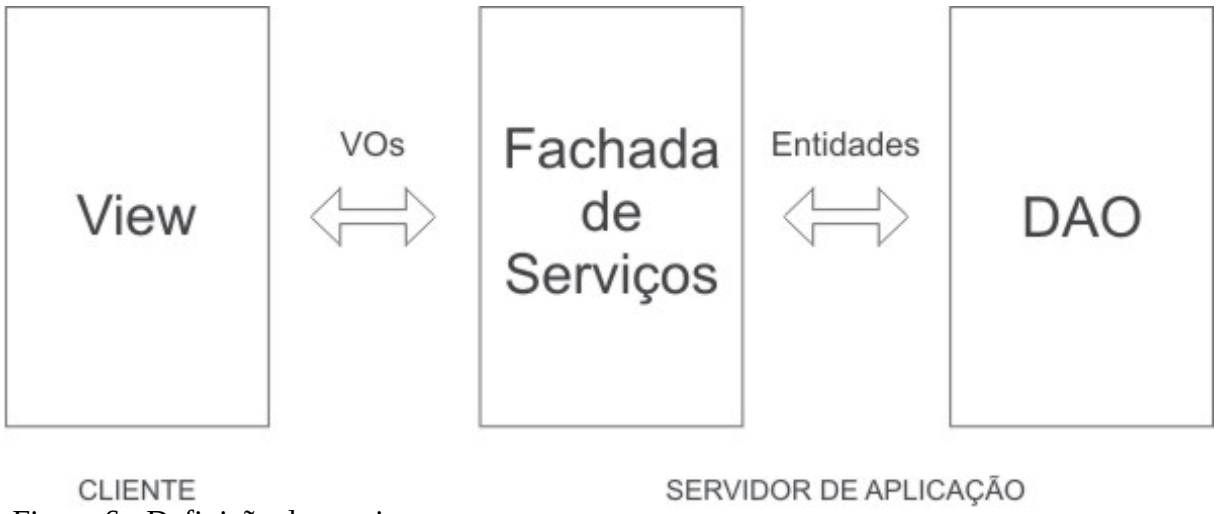

Figura 6 - Definição de arquitetura
Fonte: cedido pela empresa objeto

Conforme a figura 5, a arquitetura do sistema é composta de uma camada de apresentação, uma de fachada de serviços e outra de entidades. Além disso, contém objetos de acesso a dados e uma View Objects.

5.1.6.1. Características gerais dos projetos

Todos os projetos em estudo foram desenvolvidos na tecnologia J2EE, utilizando a ferramenta de desenvolvimento eclipse, e devem ser executados em um sistema operacional Windows. Além disso, o browser recomendado para acessá-los é o Firefox, na versão 3.0, visto que não foram projetados para serem executados em outro browser.

As equipes de análise, projeto, desenvolvimento e testes são as mesmas para os 5 projetos.

Quanto ao tamanho, todos os projetos são considerados pequenos e em geral de complexidade média.

Nas seções seguintes, são apresentados os resultados encontrados a partir da aplicação de cada técnica de estimativa descrita neste trabalho.

5.2 Wideband delphi

Inicialmente foi realizada uma reunião agendada pelo gestor de testes, onde foram apresentados o moderador das atividades e os especialistas, e designadas as tarefas de cada membro da equipe.

Foram consultados especialistas em análise de testes informando as atividades a serem estimadas, os requisitos do sistema, as condições do ambiente de teste e a base de dados inicial que ainda não estava com todos os dados necessários para os projetos 1, 2, 3 e 4. Também, foram levados em consideração fatores como a execução do *checklist* de padrões de interface gráfica bem como os casos de testes fornecidos pelo cliente.

Após duas rodadas de votação com dois especialistas nos projetos 1, 2, 3 e 5 e três rodadas no projeto 4, com 3 especialistas, seguem os resultados das estimativas realizadas pelos especialistas na tabela da página seguinte.

O primeiro projeto disponibilizado para estimar foi o projeto 4, onde participaram 3 especialistas de testes e foram necessárias 3 rodadas de votação, devido a diferença entre os valores estimados.

Somente após a terceira rodada, os especialistas chegaram a um consenso sobre o esforço necessário para a execução dos teste neste projeto.

Os projetos 1,2,3 e 5 foram disponibilizados somente um mês após a estimativa inicial realizada no projeto 4. Durante este período houve uma alteração no número de especialistas disponíveis na equipe para participar do processo de estimativas, reduzindo para dois participantes.

A tabela 1, indica na coluna “Atividades” a identificação do caso de uso que deve ser feita a estimativa de esforço, bem como o título deste, na coluna “Descrição das Atividades”. Já o nível de complexidade é indicado na coluna “Complexidade do Caso de Uso”, conforme a recomendação feita pelos analistas de sistemas do projeto. A seguir, apresenta o esforço estimado em cada rodada de votação, dividido em 3 ciclos de teste.

Finalmente, a estimativa de esforço final é apresentada, através do somatório dos valores definidos para os 3 ciclos, ao final da coluna “Estimativa média final”.

Tabela 1 - Estimativas de Especialistas

Projeto 1

Atividades	Descrição das Atividades	Complexidade do Caso de Uso (definida pelos analistas)	Estimativa média Inicial			Estimativa média 2			Estimativa média 3			Estimativa média Final		
			Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3
Esforço caso de uso 046	Gerenciar Parâmetros	Complexo * 2	10	7	10	12	7,2	12	0	0	0	12	7,2	12
Esforço caso de uso 002	Gerenciar Alíneas	Simples	3	2,1	3	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 003	Gerenciar Mensagens para Caixas Registradoras	Médio	4	2,8	4	5	3	5	0	0	0	5	3	5
Esforço caso de uso 045	Gerenciar Status de Mensagens	Simples	3	2,1	3	3	1,8	3	0	0	0	3	1,8	3
Total de estimativa média por ciclo			**20**	**14**	**20**	**23**	**13,8**	**23**	**0**	**0**	**0**	**23**	**13,8**	**23**
Total de Horas estimadas			**54**			**59,8**			**0**			**59,8**		

Projeto 2

Atividades	Descrição das Atividades	Complexidade do Caso de Uso (definida pelos analistas)	Estimativa média Inicial			Estimativa média 2			Estimativa média 3			Estimativa média Final		
			Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3
Esforço caso de uso 001	Gerenciar Cheques Devolvidos	Complexo * 3	12	8,4	12	15	9	15	0	0	0	15	9	15
Esforço caso de uso 061	Atender Solicitações de Cheques Devolvidos	Médio	2	1,4	2	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 013	Tornar Cheques Incobráveis	Médio	2	1,4	2	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 014	Lançar Bloqueios de Cheques	Complexo	4	2,8	4	5	3	5	0	0	0	5	3	5
Esforço caso de uso 041	Incluir Bloqueio Automático de Cheques Não Aceitos nas Lojas	Complexo * 2	8	5,6	8	10	6	10	0	0	0	10	6	10
Esforço caso de uso 059	Excluir/Alterar Bloqueio Automático de Cheques não Aceitos nas Lojas	Simples	1	0,7	1	1,5	0,9	1,5	0	0	0	1,5	0,9	1,5
Esforço caso de uso 011	Consultar Cheques Devolvidos	Complexo * 2	8	5,6	8	10	6	10	0	0	0	10	6	10
Esforço caso de uso 050	Consultar Estatísticas de Cheques Devolvidos e Cheques Incobráveis	Complexo	4	2,8	4	5	3	5	0	0	0	5	3	5
Esforço caso de uso 012	Consultar Cheques Bloqueados	Médio	2	1,4	2	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 006	Consultar Cheques Pendentes por Mês	Simples	1	0,7	1	1,5	0,9	1,5	0	0	0	1,5	0,9	1,5
Total de estimativa média por ciclo			**44**	**30,8**	**44**	**57**	**34,2**	**57**	**0**	**0**	**0**	**57**	**34,2**	**57**
Total de Horas estimadas			**118,8**			**148,2**			**0**			**148,2**		

Projeto 3

Atividades	Descrição das Atividades	Complexidade do Caso de Uso (definida pelos analistas)	Estimativa média Inicial			Estimativa média 2			Estimativa média 3			Estimativa média Final		
			Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3
Esforço caso de uso 042	Calcular Média p/Visamento de Cheques	Complexo * 3	12	8,4	12	15	9	15	0	0	0	15	9	15
Esforço caso de uso 016	Gerenciar CPF/CNPJ Associados por Banco, Agência e Conta	Complexo * 2	8	5,6	8	10	6	10	0	0	0	10	6	10
Esforço caso de uso 009	Consultar Cheques Recebidos por CPF/CNPJ	Complexo	4	2,8	4	5	3	5	0	0	0	5	3	5
Esforço caso de uso 018	Consultar Cheques Devolvidos por CPF/CNPJ	Médio	2	1,4	2	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 017	Gerenciar Clientes com Exceções para Visamento	Simples	1	0,7	1	1,5	0,9	1,5	0	0	0	1,5	0,9	1,5
Esforço caso de uso 051	Consultar Médias de Visamento	Médio	2	1,4	2	3	1,8	3	0	0	0	3	1,8	3
Esforço caso de uso 015	Gerenciar Exclusão CPF/CNPJ para Cheques Repetidos	Simples	1	0,7	1	1,5	0,9	1,5	0	0	0	1,5	0,9	1,5
Esforço caso de uso 019	Consultar Contas Bancárias Vinculadas por CPF/CNPJ	Simples	1	0,7	1	1,5	0,9	1,5	0	0	0	1,5	0,9	1,5
Total de estimativa média por ciclo			**31**	**21,7**	**31**	**40,5**	**24,3**	**40,5**	**0**	**0**	**0**	**40,5**	**24,3**	**40,5**
Total de Horas estimadas				**83,7**			**105,3**			**0**			**105,30**	

Projeto 4

Atividades	Descrição das Atividades	Complexidade do Caso de Uso (definida pelos analistas)	Estimativa média Inicial			Estimativa média 2			Estimativa média 3			Estimativa média Final		
			Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3
Esforço caso de uso 021	Consultar Negociações com Clientes	Simples	12	8,4	9	5	2,5	5	5	3	5	5	3	5
Esforço caso de uso 023	Registrar Negociações com Clientes	Complexo * 3	21	14,7	19	28	14	28	28	16,8	28	28	16,8	28
Esforço caso de uso 024	Efetivar Negociações com Clientes	Complexo * 2	21	14,7	18	20	10	20	20	12	20	20	12	20
Esforço caso de uso 056	Excluir Negociações Não Efetivadas	Médio	10	7	8	7	3,5	7	8,5	5,1	8,5	8,5	5,1	8,5
Esforço caso de uso 060	Informar Índices Disponíveis para Negociações	Simples	12	8,4	10	5	2,5	5	5	3	5	5	3	5
Total de estimativa média por ciclo			**76**	**53,2**	**64**	**65**	**32,5**	**65**	**66,5**	**39,9**	**66,5**	**66,5**	**39,9**	**66,5**
Total de Horas estimadas				**193,2**			**162,5**			**172,9**			**172,9**	

Projeto 5

Atividades	Descrição das Atividades	Complexidade do Caso de Uso (definida pelos analistas)	Estimativa média Inicial			Estimativa média 2			Estimativa média 3			Estimativa média Final		
			Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 1	Ciclo 2	Ciclo 3
Esforço caso de uso 034	Registrar Pagamentos na Tesouraria	Complexo	6	4,2	6	7	4,2	7	0	0	0	7	4,2	7
Esforço caso de uso 047	Consultar Cheques Cobrados na Tesouraria Geral	Médio	4	2,8	4	5	3	5	0	0	0	5	3	5
Esforço caso de uso 049	Confirmar Recebimento de Cheques da Tesouraria Geral	Médio	4	2,8	4	5	3	5	0	0	0	5	3	5
Total de estimativa média por ciclo			**14**	**9,8**	**14**	**17**	**10,2**	**17**	**0**	**0**	**0**	**17**	**10,2**	**17**
Total de Horas estimadas				**37,8**			**44,2**			**0**			**44,2**	

Fonte: Autoria própria, 2009

5.3 Pontos por casos de uso (UCP)

Para a aplicação da técnica UCP, foram utilizados os mesmos fatores de ajustes para todos os projetos, considerando a semelhança entre eles. Estes fatores foram determinados através de entrevistas com os analistas e desenvolvedores dos projetos em estudo.

Além disso, os cálculos do peso dos atores e dos casos de uso foram baseados na análise das especificações funcionais disponibilizadas.

A partir dos pontos de casos de uso calculados, foram estimados os tempos necessários para as atividades de teste levando em consideração o mesmo fator de conversão o,32, já consolidado pela fábrica ao aplicar em outras técnicas.

A tabela 2 (na página seguinte), demonstra os valores considerados para definir os fatores de ajuste (fatores técnicos e fatores ambientais). Também indica como foram calculados os fatores de ajustes dos projetos, onde obteve-se o valor total de cada fator através da multiplicação do peso associado (pré-definido pela técnica), pelo valor definido através da avaliação, considerando um intervalo entre 0 à 5. Quanto maior a atribuição do fator, maior é seu nível de influência no projeto.

Para concluir a aplicação desta metodologia, apresenta-se na tabela de classificação de atores e casos de uso, para cada projeto analisado (ver tabela 3 nas páginas 54, 55 e 56):

O cálculo do peso dos atores e dos casos de uso foram baseados na análise das especificações funcionais disponibilizadas pelo cliente. Para classificar o(s) atore(s) foi(ram) relacionado(s) o número de ator(es) com o respectivo tipo, definido por nível de complexidade. A seguir, foi multiplicado o número de atores pelo respectivo peso associado, conforme a linha “Total UAW”.

Levando em consideração o número de transações dos casos de uso, a quantidade de transações foi relacionada ao tipo de caso de uso, de acordo com a complexidade, e no final, multiplicada pelo peso associado, indicado na linha “Total UUCW”.

Com estes valores, foram calculados, primeiramente o peso total não ajustado, através do somatório entre o peso dos atores e casos de uso, apresentados na linha “UUCP = UAW + UUCW”, e posteriormente o valor total do sistema em UCP ajustados, multiplicando o peso total não ajustado pelos fatores de ajustes, conforme a linha “UCP = UUCP x TCF x EF”.

A partir dos pontos por casos de uso calculados, foram estimados os tempos necessários para as atividades de teste levando em consideração o mesmo fator de conversão o,32, já consolidado pela fábrica ao aplicar em outras técnicas.

Tabela 2 - Fatores de Ajuste

Fator	Fatores Ambientais (EF)	Peso	Avaliação(0-5)	Total
E1	Familiaridade com RUP ou outro processo	1,5	3	4,5
E2	Experiência com a aplicação em	0,5	3	1,5
E3	Experiência em Orientação a Objetos	1	3	3
E4	Presença de analista experiente	0,5	3	1,5
E5	Motivação	1	4	4
E6	Requisitos estáveis	2	3	6
E7	Desenvolvedores partime	-1	0	0
E8	Linguagem de programação difícil	-1	3	-3
	Fator de Ajuste			**17,5**
	EF = 1,4 + (-0,03 * EFactor)	**0,88**		
Fator	**Fatores de Complexidade Técnica (TCF)**	**Peso**	**Avaliação(0-5)**	**Total**
T1	Sistema distribuído	2	0	0
T2	Tempo de resposta ou taxa de transmissão	2	3	6
T3	Eficiência para usuário final	1	4	4
T4	Processamento complexo	1	0	0
T5	Código reusável	1	3	3
T6	Facilidade de instalação	0,5	4	2
T7	Facilidade de uso	0,5	4	2
T8	Portabilidade	2	3	6
T9	Facilidade de mudança (alterações)	1	3	3
T10	Concorrência	1	2	2
T11	Incluir recursos de segurança	1	3	3
T12	Acessível por terceiros	1	3	3
T13	Requer treinamento especial	1	2	2
	Fator de Ajuste			**36**
	TCF = 0,6 + (0,01 * TFactor)	**0,96**		

Fonte:Fábrica de software-RS adaptado pela autora, 2009

Tabela 3 - Classificação de atores e caso de usos

Classificação de atores (UAW)

Projeto 1		Casos de uso: Gerenciar Alíneas	Gerenciar Msgs para Caixas	Gerenciar Status de Msgs	Gerenciar Parâmetros
Tipo de ator	**Peso**	**Atores por Caso de uso**			
Simples	1	-	-	-	-
Médio	2	-	-	-	-
Complexo	3	1	1	1	1
Total UAW		**3**	**1**	**1**	**1**

Classificação de casos de uso (UUCW)

Tipos de Caso de uso	**Peso**	**Número de transações**			
Simples	1	-	-	-	-
Médio	2	-	-	-	-
Complexo	3	10	9	10	25
Total UUCW		**30**	**27**	**30**	**75**
UUCP=(UAW + UUCW)		**33**	**28**	**31**	**76**
UCP=(UUCPxTCFxEF)		**27,72**	**23,52**	**26,04**	**63,84**
TOTAL UCP Projeto1		**45,1584**			

Classificação de atores (UAW)

Projeto 2		Gerenciar Cheques Devolvidos	Consultar Cheques Pendentes por Mês	Tornar Cheques Incobráveis	Lançar Bloqueios de Cheques	Incluir Bloqueio Automática de Cheques	Excluir/Alterar Bloqueio Automática de Cheques	Consultar Cheques Devolvidos	Consultar Estatísticas Cheques Devolvidos e Incobráveis	Consultar Cheques Bloqueados	Atender Solicitações de Cheques Devolvidos
Tipo de ator	**Peso**	**Atores por Caso de uso**									
Simples	1	-	-	-	-	-	-	-	-	-	-
Médio	2	-	-	-	-	-	-	-	-	-	-
Complexo	3	1	1	1	1	3	3	2	1	3	1
Total UAW		**3**	**3**	**3**	**3**	**9**	**9**	**6**	**3**	**9**	**3**

Classificação de casos de uso (UUCW)

Tipo de Caso de uso	**Peso**	**Número de transações**									
Simples	1	-	-	-	-	-	3	-	-	-	-
Médio	2	-	4	7	-	7	-	4	7	4	4
Complexo	3	21	-	-	8	-	-	-	-	-	-
Total UUCW		**63**	**8**	**14**	**24**	**14**	**3**	**8**	**14**	**8**	**8**
UUCP=(UAW + UUCW)		**66**	**11**	**17**	**27**	**23**	**12**	**14**	**17**	**17**	**11**
UCP=(UUCPxTCFxEF)		**55,44**	**9,24**	**14,28**	**22,68**	**19,32**	**10,08**	**11,76**	**14,28**	**14,28**	**9,24**
TOTAL UCP Projeto2		**57,792**									

Classificação de atores (UAW)

Classificação de casos de uso (UUCW)

Projeto 3		Calcular Média para Visamento de Cheques	Gerenciar CPF/CNPJ Assoc. por Banco, Ag. e Conta	Consultar Cheques Recebidos por CPF/CNPJ	Consultar Cheques Devolvidos por CPF/CNPJ	Gerenciar Clientes com Exceções para Visamento	Consultar Médias de Visamento	Gerenciar Exclusão CPF/CNPJ para Cheques Repetidos	Consultar Contas Bancárias Vincul. por CPF/CNPJ
Tipo de ator	**Peso**	**Atores por Caso de uso**							
Simples	1	1	-	-	-	-	-	-	-
Médio	2	-	-	-	-	-	-	-	-
Complexo	3		3	4	5	1	1	3	5
Total UAW		**1**	**9**	**12**	**15**	**3**	**3**	**9**	**15**
Tipo de Caso de uso	**Peso**	**Número de transações**							
Simples	1	2	-	-	-	-	-	-	3
Médio	2	-	-	5	-	6	-	5	-
Complexo	3	-	10	-	8	-	8	-	-
Total UUCW		**2**	**30**	**10**	**24**	**12**	**24**	**10**	**3**
UUCP=(UAW + UUCW)		**3**	**39**	**22**	**39**	**15**	**27**	**19**	**18**
UCP=(UUCPxTCFxEF)		**2,52**	**32,76**	**18,48**	**32,76**	**12,6**	**22,68**	**15,96**	**15,12**
TOTAL UCP Projeto3		**48,9216**							

Classificação de atores (UAW)

Classificação de casos de uso (UUCW)

Projeto 4		Consultar Negociações com Clientes	Registrar Negociações com Clientes	Efetivar Negociações com Clientes	Excluir Negociações Não Efetivadas	Informar Índices Disponíveis para Negociações
Tipo de ator	**Peso**	**Atores por Caso de uso**				
Simples	1	-	-	-	1	-
Médio	2	-	-	-	-	-
Complexo	3	3	5	3	-	1
Total UAW		**9**	**15**	**9**	**1**	**3**
Tipo de Caso de uso	**Peso**	**Número de transações**				
Simples	1	3	-	-	1	2
Médio	2	-	-	-	-	-
Complexo	3	-	24	22	-	-
Total UUCW		**3**	**72**	**66**	**1**	**2**
UUCP=(UAW + UUCW)		**12**	**87**	**75**	**2**	**5**
UCP=(UUCPxTCFxEF)		**10,08**	**73,08**	**63**	**1,68**	**4,2**
TOTAL UCP Projeto4		**48,6528**				

	Projeto 5		Registrar Pagtos na Tesouraria	Consultar Chqs Cobrados na Tesouraria	Confirmar Recebimento de Chqs da Tesouraria
Classificação de atores (UAW)	**Tipo de ator**	**Peso**	**Atores por Caso de uso**		
	Simples	1	-	-	-
	Médio	2	-	-	-
	Complexo	3	1	1	3
	Total UAW		**3**	**3**	**9**
Classificação de casos de uso (UUCW)	**Tipo de Caso de uso**	**Peso**	**Número de transações**		
	Simples	1			-
	Médio	2		7	-
	Complexo	3	14	-	9
	Total UUCW		**42**	**14**	**27**
	UUCP=(UAW + UUCW)		**45**	**17**	**36**
	UCP=(UUCPxTCFxEF)		**37,8**	**14,28**	**30,24**
	TOTAL UCP Projeto5		**26,3424**		

Fonte: Autoria própria, 2009

5.4 Análise de pontos de teste (APT)

Nesta seção, apresentam-se os resultados obtidos, a partir da aplicação da técnica APT, de acordo com os dados da tabela 4 (ver tabela nas páginas 58, 59 e 60).

Conforme representado na tabela 4, foi atribuído valor igual a zero(0) aos Pontos de Teste Estáticos (PTE), pois a equipe de testes não adota processos de revisão de documentação usando *checklist*. A classificação da Qualificação da Equipe (QET) e do Ambiente de Testes (AT) foram considerados idênticos para todos os projetos, por utilizarem o mesmo nível de qualificação da equipe e configuração de ambiente.

O valor inicial para o cálculo dos Pontos de Teste Dinâmico foi o Ponto de função da função testada (PFf). Este foi fornecido pela fábrica de *software* em estudo, e encontrado pela metodologia APF. A seguir, foram atribuídos valores para os fatores: Importância do usuário(Ue), Intensidade deUso da função (Uy), Interface (I) e Complexidade (C). O grau de Complexidade (C) foi classificado com peso médio para todos os projetos, visto que a documentação necessária não está disponível aos analistas de testes.

Os valores atribuídos são somados e o resultado é dividido por 20, conforme determinado pela metodologia. O resultado encontrado para cada função foi multiplicado pelo valor de Uniformidade(U), que

mede o nível de reutilização do material de teste, e desta forma, encontrado o valor das Funções Dependentes (FDf).

As Características de Qualidade Dinâmica (QRd), foram medidas através do somatório das características explícitas Funcionalidade (F), Performance (P), Segurança (S) e Aderência e Efetividade (A), adicionadadas das Características Implícitas (CI) utilizadas. Os valores considerados para as características explícitas foram baseados no Quadro 7, e multiplicados pelo respectivo peso de cada característica, indicado pela metodologia, e dividido pelo valor 4. Enquanto que as características implícitas foram definidas com o valor 1,00 para todos os projetos levando-se em consideração somente o indicador funcionalidade (F), pois é a única característica coletada para medições quanto á qualidade dos testes.

A partir da multiplicação dos valores de Pontos de Função (PFf) pelas Funções Dependentes (FDf) e Qualidade Dinâmica (QRd), chegou-se ao valor de Pontos de Teste Dinâmico (PTDf) de cada caso de uso e posteriormente ao total destes pontos por projeto.

Após encontrar cada um destes subtotais, foi possível definir o Total dos Pontos de testes (PT), através do somatório dos Pontos de Teste Dinâmicos (PTDf), somados ao tamanho do sistema em Pontos de Função (PF) que é multiplicado pelo total de pontos de teste estaticos (PTE), considerando-se uma constante representando o valor mínimo de 500 em pontos de função. Neste caso, o único projeto que teve um valor maior para esta constante, foi o projeto 2 que apresentou o valor de 534 pontos.

Posteriormente, encontrou-se o valor da estimativa de horas de testes primárias (HTP), através da multiplicação do valor de Pontos de Testes (PT), já calculado anteriormente, pelos valores de Qualificação da Equipe (QET), definido como 1,2 e da soma dos Fatores ambientais (AT), igual a 1,1.

Finalmente, para calcular o Total de Horas de Teste (THT), foi utilizado o Indice de Planejamento e Controle (IPC), determinado com base nos Quadros 13 e 14, através do valor percentual da somas dos valores encontrados. A partir deste índice, foi adicionado ao total de horas primárias (HTP), aplicando o cálculo: THT = HTP X IPC.

Tabela 4 - Estimativas em Pontos de Teste (APT)

Ambiente de Teste(AT)	
Ferramenta de automação	4
Teste de precedência	8
Documentação de teste	3
Ambiente desenv.	2
Ambiente de teste	2
Testware	4
Total	**1,095**

PTE	
Pontos de teste estático	0
Total	**0**

QET	
Qualificação da equipe de teste	1,2
Total	**1,2**

Índice de planejamento e	
Tamanho Equipe (TE)	0,03
Ferramentas de Gerência	0,04
Total	**1,07**

Projeto 1

		Funções dependentes						Qualidade Dinâmica						
Caso de uso	**PFf**	**Ue**	**Uy**	**I**	**C**	**U**	**FDf**	**F**	**P**	**S**	**A**	**CI**	**QRd**	**PTDf**
Gerenciar Parâmetros	**120**	6	4	4	6	1,0	**1,00**	0,75	0,08	0,06	0,10	1,00	**1,99**	226,80
Gerenciar Alíneas	**18**	3	2	2	6	1,0	**0,65**	0,75	0,08	0,05	0,10	1,00	**1,98**	23,11
Gerenciar Mensagens para Caixas Registradoras	**18**	3	2	2	6	1,0	**0,65**	0,75	0,08	0,05	0,10	1,00	**1,98**	23,11
Gerenciar Status de Mensagens	**18**	3	2	2	6	1,0	**0,65**	0,75	0,08	0,05	0,10	1,00	**1,98**	23,11
Total	**174**						**2,95**						**7,91**	**296,12**
PT= PTDf+(PFxPTE)/500 =	**296,12**					**HTP = PTxQETxAT =**		**56,04**				**THT=HTP x IPC=**		**59,97**

Projeto 2

Caso de uso	PFf	Ue	Uy	I	C	U	FDf	F	P	S	A	CI	QRd	PTDf
Gerenciar Cheques Devolvidos	148	12	8	8	6	0,8	1,36	0,94	0,10	0,06	0,10	1,00	2,20	442,82
Atender Solicitações de Cheques Devolvidos	36	3	4	4	6	0,8	0,68	0,75	0,10	0,05	0,10	1,00	2,00	48,96
Tornar Cheques Incobráveis	68	3	2	4	6	0,8	0,6	0,75	0,10	0,05	0,10	1,00	2,00	81,60
Lançar Bloqueios de Cheques	62	6	2	4	6	0,8	0,72	0,75	0,10	0,05	0,10	1,00	2,00	89,28
Incluir Bloqueio Automático de Cheques	44	3	4	4	6	0,8	0,68	0,75	0,10	0,05	0,10	1,00	2,00	59,84
Excluir/Alterar Bloqueio Automático de Cheques	28	6	4	4	6	0,8	0,8	0,75	0,10	0,05	0,10	1,00	2,00	44,80
Consultar Cheques Devolvidos	68	3	4	2	6	0,8	0,6	0,75	0,10	0,05	0,10	1,00	2,00	81,60
Consultar Estatísticas de Cheques Devolvidos e Incobráveis	52	3	2	2	6	0,8	0,52	0,75	0,10	0,05	0,10	1,00	2,00	54,08
Consultar Cheques Bloqueados	28	3	4	2	6	0,8	0,6	0,75	0,10	0,05	0,10	1,00	2,00	33,60
Consultar Cheques Pendentes por Mês	20	3	2	2	6	1,0	0,65	0,75	0,10	0,05	0,10	1,00	2,00	26,00
Total	534						7,21						20,20	962,58
PT= PTDf+(PFxPTE)/534 =	**962,58**			**HTP = PTxQETxAT =**				**182,17**				**THT=HTP x IPC=**		**194,93**

Projeto 3

Caso de uso	PFf	Ue	Uy	I	C	U	FDf	F	P	S	A	CI	QRd	PTDf
Calcular Média para Visamento de Cheques	18	12	2	8	6	1,0	1,4	0,94	0,10	0,05	0,10	1,00	2,19	55,13
Gerenciar CPF/CNPJ Associados por Banco, Agência e Conta	82	6	4	4	6	1,0	1	0,94	0,10	0,05	0,10	1,00	2,19	179,38
Consultar Cheques Recebidos por CPF/CNPJ	30	6	4	2	6	1,0	0,9	0,75	0,10	0,05	0,10	1,00	2,00	54,00
Consultar Cheques Devolvidos por CPF/CNPJ	80	3	4	2	6	1,0	0,75	0,75	0,10	0,05	0,10	1,00	2,00	120,00
Gerenciar Clientes com Exceções para Visamento	14	6	4	2	6	1,0	0,9	0,94	0,10	0,05	0,10	1,00	2,19	27,56
Consultar Médias de Visamento	24	3	2	2	6	1,0	0,65	0,75	0,10	0,05	0,10	1,00	2,00	31,20
Gerenciar Exclusão CPF/CNPJ p/ Cheques Repetidos	20	3	4	4	6	1,0	0,85	0,75	0,10	0,05	0,10	1,00	2,00	34,00
Consultar Contas Bancárias Vinculadas por CPF/CNPJ	20	3	4	2	6	1,0	0,75	0,75	0,10	0,05	0,10	1,00	2,00	30,00
Total	288						7,2						16,56	531,26
PT= PTDf+(PFxPTE)/500 =	**531,26**			**HTP = PTxQETxAT =**				**100,55**				**THT=HTP x IPC=**		**107,58**

Projeto 4

Caso de uso	PFf	Ue	Uy	I	C	U	FDf	F	P	S	A	CI	QRd	PTDf
Consultar Negociações com Clientes	74	3	4	2	6	1,0	0,75	0,75	0,10	0,06	0,10	1,00	2,01	111,69
Registrar Negociações com Clientes	182	6	8	4	6	1,0	1,2	1,13	0,10	0,06	0,10	1,00	2,39	521,43
Efetivar Negociações com Clientes	164	6	4	4	6	1,0	1	1,13	0,10	0,06	0,10	1,00	2,39	391,55
Excluir Negociações Não Efetivadas	16	3	4	4	6	1,0	0,85	0,75	0,10	0,06	0,10	1,00	2,01	27,37
Informar Índices Disponíveis p/ Negociações	14	3	2	2	6	1,0	0,65	0,75	0,10	0,06	0,10	1,00	2,01	18,31
Total	450						4,45						10,81	1070,36
PT= PTDf+(PFxPTE)/500 =	1070,36			HTP = PTxQETxAT =				R$ 202,57				THT=HTP x IPC=		216,75

Projeto 5

Caso de uso	PFf	Ue	Uy	I	C	U	FDf	F	P	S	A	CI	QRd	PTDf
Registrar Pagamentos na Tesouraria	68	6	4	4	6	1,0	1	0,75	0,10	0,06	0,10	1,00	2,01	136,85
Consultar Cheques Cobrados na Tesouraria	62	3	2	2	6	1,0	0,65	0,75	0,10	0,06	0,10	1,00	2,01	81,10
Confirmar Recebimento de Cheques da Tesouraria	58	3	4	4	6	1,0	0,85	0,75	0,10	0,06	0,10	1,00	2,01	99,22
Total	188						2,5						2,01	317,17
PT= PTDf+(PFxPTE)/500 =	317,17			HTP = PTxQETxAT =				60,03				THT=HTP x IPC=		64,23

Fonte: Autoria própria, 2009.

5.5 Análise de pontos de função (APF)

Os dados com as estimativas desta técnica foram fornecidos pela empresa, visto que já utilizam na fábrica em estudo (ver ANEXO D). O fator de conversão aplicado em horas, para as atividades de teste é de 0,32, conforme mencionado anteriormente.

5.6 Esforço realizado

Após a realização dos cálculos, de acordo com as técnicas descritas neste projeto, além de considerar o método APF, já aplicado pela empresa em estudo, foram feitas comparações com os esforços realizados fornecidos pela empresa, conforme a tabela a seguir:

Tabela 5 – Esforço realizado

Projetos	ESTIMATIVAS				Esforço realizado	Desvio padrão
	APF	Wideband	APT	UCP		
Projeto 1	55,68	59,80	59,97	45,16	**78,65**	17,68
Projeto 2	177,28	148,20	194,93	57,79	**211,32**	15,39
Projeto 3	92,16	105,30	107,58	48,92	**226,85**	118,27
Projeto 4	144,00	172,90	216,76	48,65	**324,36**	106,60
Projeto 5	60,16	44,20	64,23	26,34	**66,00**	0,77
Total	**529,28**	**530,4**	**643,47**	**226,86**	**907,18**	**262,71**

Fonte: Autoria própria, 2009.

5.6.1 Análise dos resultados

O porte dos projetos analisados foi classificado como médio, não sendo possível comparar para efeito de métricas.

O maior desvio ocorreu nos projetos 3 e 4. Nos demais foram considerados desvios baixos. Enquanto que o menor desvio observado, foi no projeto 5, que pode ser considerado nulo.

Observou-se que a técnica Análise de Pontos de Teste (APT) foi a que mais se aproximou dos resultados finais de esforço realizado em todos os projetos. No entanto, o desvio nos proje-

tos 3 e 4 ainda foi considerado alto, mesmo comparando com esta técnica.

A partir da análise dos resultados e consulta aos gestores e relatórios dos projetos, os fatores relacionados a seguir podem ter influenciado no resultado final:

Projeto 1 - Os testes do caso de uso “Gerenciar Parâmetros” foram realizados em apenas umciclo de testes completo e não foram re-testados todos os defeitos encontrados no segundo ciclo. Isto ocorreu devido ao atraso na entrega pela equipe de desenvolvimento, gerando problemas na qualidade do produto. Além disso, foram encontradas muitas inconsistências na documentação de requisitos.

Projeto 2 - Ocorreu o maior número de inconsistências nos documentos de requisitos funcionais, gerando retrabalho para as equipes de análise, desenvolvimento e testes.

Projeto 3 - Nos casos de uso “Calcular média para Visamento de cheques” e “Gerenciar CPF/CNPJ Associados por Banco, Agência e Conta”, foram registrados atrasos na entrega pelo desenvolvimento, sendo liberados em partes durante a execução dos testes, fazendo com que fosse necessária uma regressão de testes já executados nos testes anteriores.

Projeto 4 - Os casos de uso “Registrar negociação”e “Efetivar Negociação", foram entregues em três etapas para a equipe de testes, ou seja, estava em desenvolvimento durante a fase de testes. Isto comprometeu o esforço realizado no ciclo inicial e foram necessários mais de 3 ciclos, conforme recomendado para o projeto.

Projeto 5 - Os testes foram realizados em um ambiente separado do desenvolvimento e a base de dados inicial foi fornecida adequadamente, antes do início dos testes. Além disso, o *checklist* de interface foi utilizado somente no 1º ciclo de testes. A duração da execução de testes em todos os projetos mencionados foi de 3 ciclos em média.

Outro fator que deve ser levado em consideração, para todos os projetos, foi o caso de serem encontradas inconsistências nos requisitos funcionais, durante a execução dos testes.

Isto gerou nova consulta aos analistas, e, na maioria das vezes, alterações nos requisitos

iniciais já desenvolvidos. Consequentemente, tornou-se necessário um aumento no esforço, envolvendo as equipes de desenvolvimento e testes, que não havia sido considerado nas estimativas iniciais.

Além disso, os testes dos projetos 1, 2, 3 e 4 foram executados no mesmo ambiente de desenvolvimento, sendo necessário que os testadores tivessem o sistema instalado localmente em cada máquina, sendo necessário um tempo adicional cada vez que fosse necessário atualizar o ambiente.

A base de dados inicial, fornecida pelo cliente apresentou falta de dados essenciais para a execução dos testes, fazendo com que os testadores juntamente com a equipe de desenvolvimento inserissem manualmente alguns dados para que fosse possível o início das atividades.

É importante mencionar que alguns defeitos encontrados foram identificados como sendo de um *framework* que é construído e mantido por uma equipe de desenvolvimento no cliente. Por isso, os esforços de correções de defeitos e teste destes erros não fizeram parte das métricas deste trabalho.

6 CONCLUSÕES E TRABALHOS FUTUROS

Existe uma grande necessidade de aperfeiçoar as estimativas de esforço em projetos de testes de *software*. Estas estimativas são importantes para o gerenciamento durante todo o projeto.

Atualmente, existem diversas técnicas de estimar que vem sendo utilizadas em projetos de desenvolvimento de *software*, como Pontos por casos de uso (UCP), Análise de pontos de teste(APT), Análise de pontos de função (APF) e Wideband delphi. Porém, existem poucas experiências relacionadas à aplicação de técnicas de estimar em projetos de testes.

Neste trabalho, as metodologias citadas foram aplicadas em cinco projetos de testes em uma fábrica de *software*, com o objetivo de avaliar a sua utilização e melhorar a precisão das estimativas nestes projetos.

Após os resultados obtidos, pode-se considerar que o atraso pela equipe de desenvolvimento, inconsistências nos documentos de especificações funcionais, a falta de uma base inicial de dados, bem como o ambiente em que os testes são executados, influenciam diretamente no esforço de um projeto de testes. Além disso, nenhum método é melhor do que outro em todos os aspectos, por isso faz-se necessário adaptá-los de acordo com a necessidade e o andamento de cada projeto.

Embora a estimativa total apresentada pela técnica Análise de Pontos de Teste, foi a que mais se aproximou do esforço realizado, sua aplicação não teve resultados satisfatórios, uma vez

que as estimativas obtidas ainda foram subestimadas com relação ao tempo consumido na prática. Portanto, recomenda-se continuar este estudo em outros projetos a fim de chegar a técnica mais adequada para cada tipo de projeto.

Considera-se que foi atingido o objetivo deste estudo, quanto a realizar experimentos de estimativas em projetos de testes, utilizando dados reais. Desta forma, puderam ser validadas técnicas bem sedimentadas em projetos de testes, servindo de base para futuras estimativas de esforço.

6.1 Dificuldades encontradas

Ao realizar esse estudo, destaca-se algumas dificuldades que limitaram seu desenvolvimento. A primeira delas diz respeito a divergências encontradas na literatura sobre a aplicação de algumas técnicas estudadas. Além disso, foram encontrados poucos experimentos de estimativas de esforço em projetos de teste.A maioria dos autores foca na aplicação em projetos de desenvolvimento de *software*.

Outra dificuldade encontrada, foi a demora na obtenção dos resultados realizados efetivamente nos projetos em estudo, para a avaliação das estimativas encontradas.

Por fim, pode-se considerar dificuldades, as limitações que a própria estrutura do trabalho impõe, em relação ao prazo e ao tamanho do estudo.

6.2 Trabalhos Futuros

A proposta inicial, dando continuidade a este trabalho, é a aplicação das metodologias utilizadas, em projetos com portes diferentes, a fim de focar nesta métrica para a comparação de estimativas de esforço.

Outra proposta para a ampliação deste estudo é a aplicação de uma mescla de técnicas de estimativas, para utilizar os recursos que mais se adaptem às atividades de teste.

Finalmente torna-se interessante a realização das técnicas deste estudo, levando em consideração projetos de teste que envolvam todas as etapas do processo: procedimentos iniciais, planejamento, preparação, especificação, execução e entrega. Desta forma, será possível identificar quando é mais vantajoso usar uma técnica em detrimento de outra.

REFERÊNCIAS

A GUIDE to the project management body of knowledge: (PMBOK guide). 3rd ed. Newtown Square [Estados Unidos]: Project Management Institute, 2004.

BOEHM, Barry. W. **Software engineering economics.** New Jersey: Prentice-Hall, 1981.

DELAMARO, Márcio. **Introdução ao teste de software.** Rio de Janeiro: Elsevier: Campus, 2007

GINDRI, Vanessa, **Teste de Regressão**. São Paulo: Instituto de Computação, UNICAMP, 2001. Disponível em: <http//:www.ic.unicamp.br/~eliane/Cursos/ SeminarioTestes/ Teste_Regressao.ppt>. Acesso em: 16 ago 2008

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. Computer Society; Standard 829: Standard for Software Test Documentation; Nova York:IEEE, 1988.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION / INTERNATIONAL ENGENEERING CONSORTIUM 9126, **Software engineering** – Product quality – Part 2: external metrics, 2001.

KARNER, G., **Use Case Points - Resource Estimation for Objectory Projects, Objective Systems SF AB.** University of Linköping, Suécia, 1993.

MONTEIRO, Tânia Cavalcanti. **Uma Extensão de Estimativas baseadas em UCP Atendendo às Necessidades do CMMISW Nível 2 e 3**. 2004. Dissertação de Mestrado, UNIFOR, Brasil. Disponível em: <http//www.sbc.org.br/bibliotecadigital/download.php?paper=686>. Acessado em: 21 jul 2008

NAGESWARAN, S. **Test Effort Estimation Using Use Case Points.** Em 14th International Internet Software Quality Week 2001, Junho, São Francisco. Disponível em: <http:// www.cognizant.com/html/content/cogcommunity/Test_Effort_Estimation.pdf >. Acessado em: 02 jun 2008.

PAULA FILHO, Wilson P., **Engenharia de software:** fundamentos métodos e padrões. Rio de Janeiro: LTC, 2003.

PEREIRA, R., TAIT, T. F. C., Castro, C. Y. H., Trindade, D. F. G., Silva, J. V., **Estimativas de Software: O estudo de uma aplicação prática utilizando a técnica de use case points**. Maringá, 2007.

PETERS, James F; PEDRYCZ, Witold. **Engenharia de software:** teoria e prática. Rio de Janeiro: Campus, 2001.

PEZZÈ, Mauro, YOUNG, Michal, **Teste e Análise de Software – processos, princípios e técnicas.** Porto Alegre, Bookman, 2008.

POL, M., TEUNISSEN, R. e VEENENDAAL, E. van., **Software Testing** - A Guide to the Tmap Approach. s.1. Addison Wesley. IFPUG. *Counting Practices Manual*. Version 4.2, 2004. www.ifpug.org

PRESSMAN, Roger S. **Engenharia de software.** São Paulo: Makron Books, 1995.

PRESSMAN, Roger S.. **Software engineering:** a practitioner's approach. 6th ed. New York: McGraw-Hill, 2005.

RIOS, Emerson. **Análise de riscos em projetos de teste de software.** Rio de Janeiro: Alta Books, 2005.

RIOS, Emerson; MOREIRA FILHO, Trayahú R. **Teste de software.** 2. ed., rev. e ampl. Rio de Janeiro: Alta Books, 2003.

RIOS, Emerson. e MOREIRA FILHO, Trayahú. R., **Base de Conhecimento em Teste de Software.** Rio de Janeiro, Martins Fontes, 2007.

ROCHA, Ana. R. C., MALDONADO, José. C. e WEBER, Kival C., **Qualidade de Software: Teoria e Prática.** São Paulo, Prentice-Hall, 2001.

SOMMERVILLE, Ian. **Engenharia de software.** 6. ed. São Paulo: Addison-Wesley, 2003.

SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 3., 2004 maio 31-jun. 4, Brasília, DF; OLIVEIRA, Káthia Marçal de; WEBER, Kival Chaves (Coord.) **Anais ...** Brasília, DF: 2004.

UNITED KINGDOM SISTEM METRICS ASSOCIATION. **MK II Function Point Analysis Counting Pratices Manual:** version 1.3.1. Technical report, London: United Kingdom Software Metrics Association, 1998.

VASQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado. **Análise de pontos de função:** medição, estimativas e gerenciamento de projetos de software. 3. ed., rev., atual. e ampl. São Paulo: Érica, 2005.

ANEXO A - Checklist de padrões

Versão do Checklist: 2.11 | Versão do Padrão de Interface: 1.0.31					
Checklist de Teste Exploratório – UCXXX – Nome da tela – Sistema: XXX					
Nº	**Item**	**Release**	**Descrição**	**Resultados**	**Observação**
			Comportamento Telas em Geral		
1	**Fechamento**	**Todas**	Verificar se além dos meios tradicionais, é possível fechar a tela pressionando a tecla ESC.		
2	**Foco padrão**	**Todas**	Assim que a tela é aberta o foco deve estar no primeiro campo a esquerda e acima que seja editável ou selecionável (combo).		
3	**Limpar**	**Todas**	Ao disparar a ação "limpar a tela" deve-se retornar ao estado inicial da tela conforme EFR, com exceção das colunas, caso tenham sido trocadas de lugar.		
4	**Consulta Fonética**	**Todas**	Telas que possuam consulta fonética devem seguir o Algoritmo de		
5	**Ortografia**	**Todas**	Verificar ortografia dos campos e labels da tela.		
6		**Todas**	Os nomes dos campos sempre devem trazer a primeira letra da primeira palavra em maiúsculo e demais em minúsculo. A exceção são os campos que possuem o nome separado por barra "/", onde a palavra depois da barra também deve ser maiúscula, exemplo: Sistema/Subsistema.		
7	**Teclas de Atalho**	**Todas**	As teclas de atalho F1-F10 que estejam ativadas, devem funcionar imediatamente após acionadas.		
8	**Resolução**	**Todas**	A aplicação deve comportar uma resolução de 1024 X 768.		
9	**Tela de Ajuda**	**Todas**	Ao disparar-se a ação de Ajuda, sistema deve apresentar uma tela de Ajuda sobre a tela em questão. Ao posicionar-se o mouse sobre qualquer um dos botões, deve ser exibido um tooltip informativo sobre as finalidades daquele botão.		
10	**Capitalização**	**NENHUMA**	Para toda a aplicação WEB, conforme é digitado qualquer texto, o sistema deve ir transformando e exibindo os caracteres em letras MAIÚSCULAS.	**N/A**	
11		**NENHUMA**	A exceção para esta regra são os campos TextArea onde esta regra não se aplica. Nesses componentes é possível digitar-se caracteres proibidos, maiúsculas, minúsculas, etc.	**N/A**	
12	**Abertura de Telas**	**Todas**	Telas de Migração (chamadas através do botão "Migração") devem ser iniciadas na mesma janela da tela chamadora. Deve possuir o Barra de Navegação (ex.: Pessoas > Endereço > etc).		
13		**Todas**	Telas de Captura (chamadas através do botão [...]) e de Listar (através do botão Listar) devem ser iniciadas em popups. Não devem possuir Barra de Navegação.		
			Comportamento Telas de Cadastro		
14	**Condição inicial das grades**	**Todas**	Telas de Cadastro NÃO devem iniciar preenchidas, a menos que seja indicado na EFR o contrário.		
15	**Inclusão**	**Todas**	Após inclusão o sistema deve: - atualizar a grade exibindo APENAS o registro incluído; - limpar os campos da tela; - posicionar o cursor no primeiro campo editável ou selecionável da tela.		
16	**Alteração**	**Todas**	Após alteração, o sistema deve: - atualizar a grade APENAS com o registro recém alterado; - atualizar os campos da tela.		
17	**Exclusão**	**Todas**	Após a exclusão, o sistema deve: - atualizar a grade retirando o registro excluído da mesma; - limpar os campos da tela; - posicionar o cursor no primeiro campo editável ou selecionável da tela.		
18	**Consulta**	**Todas**	Os campos pesquisáveis deixados em branco devem ser desconsiderados. Sistema deve retornar registros na grade de acordo com o informado nos campos filtros. O cursor deve ficar posicionado no primeiro campo para edição ou seleção.		
19		**Todas**	**LISTAR** O comportamento fortemente indicado para telas que não possuam grade mas possuam a opção Listar deve ser o seguinte: para um consulta exata, comportamento normal; para mais de um resultado na consulta, disparar a ação Listar AUTOMATICAMENTE assumindo como filtro os dados informados na tela chamadora.		
20	**Selecionar na Grade**	**Todas**	Ao selecionar-se algum registro na grade, o sistema deve carregar os dados do registro selecionado, preenchendo os campos da tela com esses dados.		
			Comportamento Telas de Movimento		
21	**Condição inicial das grades**	**Todas**	Telas de Movimento NÃO devem iniciar preenchidas, a menos que seja indicado na EFR o contrário.		
22	**Inclusão**	**Todas**	Após inclusão o sistema deve: - atualizar a grade ADICIONANDO aos registros da grade o recém incluído; - limpar os campos da tela; - posicionar o cursor no primeiro campo editável ou selecionável da tela.		

ANEXO B - Caso de Teste

ID 13763 :: Caso de Teste 024 - Efetivar negociação Versão 2 Sumário:

Histórico da ETF:

16/09/2008 - Criação da ETF conforme funcional 2.00.

03/10/2008 - Alteração do CT referente a issue 5049 conforme funcional 2.01.

08/12/2008 - Atualização das ETFs para a versão 2.04 conforme issues 5499, 5500 e 5983.

1.Pré Condições

1.1. Usuário deve ter permissão de acesso a Tesouraria Loja.

1.2. Usuário deve estar autenticado no sistema e possuir permissões de acesso a tela de Efetivar Negociações com Clientes.

1.3. Usuário deve estar autenticado no sistema e possuir permissões de acesso a tela de Confirmação de Recebimentos CHQ/ CRE/ SHP.

1.4. Usuário deve visualizar o menu principal do sistema de Cheques.

1.5. Usuário deve visualizar o menu principal do sistema de Tesouraria.

1.6. Negociação de Cliente (CPF/CNPJ) deve possuir 1 (uma) negociação cadastrada no sistema com situação “Não Efetivada” e “Efetivada” com data de efetivação igual a data atual e as formas de pagamento e o turno não devem ter sido atualizado pelo sistema de Tesouraria.

Cobertura:

Subfluxo Consultar Negociação

Subfluxo Efetivar Negociação

[RN19] Efetivação da Negociação

[RN17] Agência e Conta do Cheque

[RN23] Local para resgate do cheque

[RN24] Data para resgate do cheque

[RN28] Habilitação dos botões (opção I)

[RN30] Relatório Contábil

Passos: Resultados Esperados:

1. Logar no Sistema de Cheques em um Local diferente de ADM.

2. Clicar na opção “Efetivar Negociações com Clientes” do menu “Movimento” do sistema de Cheques.

ANEXO C - Casos de Uso

1 Descrição

Este caso de uso descreve o processo de carga automática dos arquivos de retorno de cobrança das empresas externas. É um caso de uso executado pelo sistema, sem participação direta de nenhum usuário. Este caso de uso não tem como objetivo a confirmação do recebimento, isso é feito no UC40

2 Diagrama do Caso de Uso

2.1 Atores

2.2 Atividades e Atores

2.1.1 System Clock

2.3 Diagrama de Estados

3 Interface

Esse caso de uso descreve um processo interno, não sendo utilizada tela.

4 Fluxo de Eventos

Os fluxos de eventos representam os passos executados pelo sistema ou pelo usuário para a execução do caso de uso.

4.1 Pré-condições

As pré-condições definem as condições que devem estar atendidas para que o caso de uso possa ser executado.

4.2 Fluxo Básico

4.2.1 O Ator lê os arquivos de retorno [RN01] [RN04], com o nome externo de DEVhnnn.txt e disponibiliza-os para confirmação (vide RN03)." [RN02]

4.2.2 O Ator renomeia o arquivo de DEVhnnn.txt para RETnnnn.txt [RN01].

ANEXO D - Planilha de Estimativas

Fábrica de Software	Planilha de Estimativas	Versão do Template:
	Projetos	#REF!

					Grupo 1								
Projeto	Código	Caso de Uso	Versão EFR	Compl.	Campos NormaisTela	Campos a + Grade	Campos do Relatório	Peso não ajustado	% Reuso	Ajuste	Peso Ajustado	Factory Points não ajustados	Factory Points
1	UC046	Gerenciar Parâmetros	2.06	Complexo	28	1	0	16	0%	0	16	120	120
1	UC002	Gerenciar Alíneas	2.03	Simples	2	0	0	2	0%	0	2	18	18
1	UC003	Gerenciar Mensagens para Caixas Registradoras	2.01	Simples	2	0	0	2	0%	0	2	18	18
1	UC045	Gerenciar Status de Mensagens	2.03	Simples	2	0	0	2	0%	0	2	18	18
2	UC001	Gerenciar Cheques Devolvidos	2.07	Complexo * 3	29	0	0	16	0%	0	16	148	148
2	UC061	Atender Solicitações de Cheques Devolvidos	2.03	Médio	12	9	0	8	0%	0	8	36	36
2	UC013	Tornar Cheques Incobráveis	2.03	Simples	11	22	22	32	0%	0	32	68	68
2	UC014	Lançar Bloqueios de Cheques	2.01	Complexo	13	1	0	4	0%	0	4	62	62
2	UC041	Incluir Bloqueio Automático de Cheques Não Aceitos nas Lojas	2.04	Complexo	14	3	0	8	0%	0	8	44	44
2	UC059	Excluir/Alterar Bloqueio Automático de Cheques não Aceitos nas Lojas	2.02	Simples	12	0	0	4	0%	0	4	28	28
2	UC011	Consultar Cheques Devolvidos	2.05	Complexo * 2	6	26	17	32	0%	0	32	68	68
2	UC050	Consultar Estatísticas de Cheques Devolvidos e Cheques Incobráveis	2.03	Complexo	13	9	21	32	0%	0	32	52	52
2	UC012	Consultar Cheques Bloqueados	2.01	Médio	8	8	0	4	0%	0	4	28	28
2	UC006	Consultar Cheques Pendentes por Mês	2.03	Simples	7	3	0	4	0%	0	4	20	20
3	UC042	Calcular Média para Visamento de Cheques	2.03	Complexo * 3	0	0	0	0	0%	0	0	18	18
3	UC016	Gerenciar CPF/CNPJ Associados por Banco, Agência e Conta	2.08	Complexo * 2	14	4	0	8	0%	0	8	82	82
3	UC009	Consultar Cheques Recebidos por CPF/CNPJ	2.02	Complexo	6	13	0	8	0%	0	8	30	30
3	UC018	Consultar Cheques Devolvidos por CPF/CNPJ	2.04	Médio	7	35	20	32	0%	0	32	80	80
3	UC017	Gerenciar Clientes com Exceções para Visamento	2.02	Simples	4	0	0	2	0%	0	2	14	14
3	UC051	Consultar Médias de Visamento	2.02	Médio	3	3	0	2	0%	0	2	24	24
3	UC015	Gerenciar Exclusão CPF/CNPJ para Cheques Repetidos	2.01	Simples	2	0	0	2	0%	0	2	20	20
3	UC019	Consultar Contas Bancárias Vinculadas por CPF/CNPJ	2.02	Simples	8	3	0	4	0%	0	4	20	20
4	UC060	Informar Índices Disponíveis para Negociações	2.02	Simples	1	4	0	2	0%	0	2	14	14