UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E DE ESTATÍSTICA

CURSO DE PÓS- GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

COMPARAÇÃO E AVALIAÇÃO ENTRE O PROCESSO RUP DE DESENVOLVIMENTO DE SOFTWARE E A METODOLOGIA EXTREME PROGRAMMING

Dissertação de Mestrado submetido à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Ciência da Computação.

Marcos Leandro Nonemacher

FLORIANÓPOLIS, ABRIL DE 2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E DE ESTATÍSTICA

CURSO DE PÓS- GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

COMPARAÇÃO E AVALIAÇÃO ENTRE O PROCESSO RUP DE DESENVOLVIMENTO DE SOFTWARE E A METODOLOGIA EXTREME PROGRAMMING

Marcos Leandro Nonemacher

Dissertação de Mestrado submetida à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora

Prof.ª Anita Maria da Rocha Fernandes, Drª

FLORIANÓPOLIS, ABRIL DE 2003

COMPARAÇÃO E AVALIAÇÃO ENTRE O PROCESSO RUP DE DESENVOLVIMENTO DE SOFTWARE E A METODOLOGIA EXTREME PROGRAMMING

Marcos Leandro Nonemacher

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação no Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Florianópolis, 16 de abril de 2003.

_____

Prof. Dr. Fernando Ostuni Gauthier

Banca Examinadora:

_____

Profª . Drª. Anita Maria da Rocha Fernandes

(Orientadora)

_____

Prof. Dr. José Mazzucco Júnior

_____

Prof. Dr. Bruno Mazzola

Dedico este trabalho a minha esposa e a meus filhos que apesar de ainda estarem no ventre da mãe já fazem parte das nossas vidas.

Agradeço a Deus, por me dar a vida e todos os sentidos perfeitos para conseguir vencer mais esta etapa, a meus pais pelo apoio e incentivo, aos colegas Elcio, Maciel, Rosangela, Ivani, Tiago, Paulo, Ruben, Juliana e Soeli, que participaram efetivamente para a realização desta pesquisa e a orientadora Anita que além da orientação em alguns momentos teve papel de mãe que ao mesmo tempo em que exigia, incentivava para a conclusão deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE REDUÇÕES

UML	Unified Model Language, Linguagem de Modelagem Unificada
XP	Extreme Programmig
RUP	Rational Unified Process
EUA	Estados Unidos da América

RESUMO

O objetivo desta pesquisa é comparar e avaliar a metodologia Extreme Programming – XP e o processo RUP de desenvolvimento de software. Tanto XP quanto RUP são metodologias recentes, a primeira no entanto faz parte de um grupo de metodologias ditas como ágeis, já a segunda é o refinamento de outras metodologias orientadas a objetos, que contempla dentre as suas inúmeras técnicas e práticas, a criação de modelos e diagramas. Várias destas práticas são contraditórias entre as duas metodologias, devido a isto a motivação deste trabalho em aplicar as metodologias na criação de um produto de software, com os mesmos requisitos e escopo de projeto para que pudesse ser comparado o resultado final. Como conclusão da pesquisa, identificou-se que a XP obteve um maior custo/benefício, tendo como base que o esforço de tempo necessário para o desenvolvimento e testes foi praticamente o mesmo, produzindo um resultado final de menor qualidade. Os dados resultantes da pesquisa levam a concluir que a utilização da metodologia RUP seria mais adequada, levando-se em consideração que o objetivo era saber qual das duas metodologias proporciona o desenvolvimento de um software de melhor qualidade, com o menor custo possível.

ABSTRACT

The objective of this research is to compare and evaluate the Extreme Programming – XP methodology and the process RUP of development of software. Either XP or RUP are recent methodologies, the first however belongs to a group of methodologies said as agile, yet the second is the evolution of other methodologies oriented to objects, that contemplate inside its countless techniques and practical, the creation of models and diagrams. A series of these practice are contradictories between the both methodologies, due to this motivation of this work in applying the methodologies in the creation of a new software product, with the same requisite and structure of projects so that it could be compared the final result. As conclusion of the research, if identified that the XP obtained a lesser benefit cost, having as base that the efforts of time are necessary for the development and tests were practically the same and produced a final result of lesser quality. The resulted datum of research take to conclude that the utilization of RUP methodology would be appropriate, taking in consideration that the objective was to know which of the both methodologies provide the development of a better quality software, with the lesser possible cost.

CAPÍTULO I – INTRODUÇÃO

1.1 Apresentação

A rápida evolução e abrangência dos sistemas de informação no decorrer dos últimos 40 anos, confirmam a importância dos mesmos como sendo a alavanca das mudanças significativas no modelo de negócio das empresas, proporcionando para as organizações a obtenção de vantagens competitivas e estratégicas.

O software, por ser um dos componentes de maior importância dos sistemas de informação, vem sendo alvo de constantes estudos objetivando ter um desenvolvimento de maneira previsível e em determinado período, com utilização eficiente e eficaz dos recursos.

Um fator chave para se alcançar estes objetivos é adoção de uma metodologia para o desenvolvimento de software. Uma metodologia de desenvolvimento é um conjunto de regras e padrões que orientam a abordagem utilizada em todas as tarefas do ciclo de desenvolvimento, prescrevendo ferramentas e até como elas devem ser utilizadas (STAIR,1998).

Inúmeras metodologias já surgiram, e certamente novas serão desenvolvidas. Algumas são mais utilizadas e conhecidas do que outras, porém o emprego das metodologias mais conhecidas pode trazer alguns benefícios para as

empresas, mas não garante necessariamente que por serem mais difundidas sejam as mais eficientes e eficazes.

A metodologia de desenvolvimento de software impõe um processo disciplinado sobre o desenvolvimento do mesmo com o objetivo de fazê-lo mais previsível e eficiente. Algumas fazem isso através do desenvolvimento de um processo detalhado, com uma forte ênfase no planejamento, inspiradas por áreas da engenharia. A crítica sobre elas é que são muito burocráticas, sendo necessária a produção de muito material para ser possível seguí-las. Devido a isto freqüentemente são chamadas de “peso-pesado” ou metodologias monumentais.

Em reação às “peso-pesado” surgiu nos últimos anos um novo grupo de metodologias, denominadas por um tempo de “peso-leve”, mas agora conhecidas por “metodologias ágeis”. Diferentemente das peso-pesado, as metodologias peso-leve providenciam apenas o processo suficiente para se obter um retorno adequado, são menos burocráticas e menos formais, no entanto, essa informalidade poderá ocasionar situações onde a falta de uniformidade esteja mais intrínseca (HIGHSMITH,2001).

Como conseqüência desta contradição de idéias abre-se espaço para pesquisas com intuito de amadurecer e refinar o que cada metodologia apresenta de positivo. O objeto de estudo deste trabalho será comparar as metodologias *RUP (Rational Unified Process) e XP (Extreme programming*) .

1.2 Justificativa

O desenvolvimento de software ainda é algo novo com muito a ser pesquisado e explorado; a engenharia de software comparada com outros ramos das engenharias é incipiente. Neste processo de amadurecimento muitas filosofias e correntes de pensamento aparecem, dentre elas um grupo que defende as Metodologias Ágeis de Desenvolvimento e outros defendem Metodologias mais Clássicas ou Tradicionais.

Ainda é um desafio para engenharia de software adaptar métodos convencionais a técnicas evolucionárias de trabalho, rápidas e extremas (FINKELSTEIN e KRAMER, 2002).

1.3 Importância

A importância deste estudo está no fato de contribuir para a comunidade de desenvolvimento de software, apresentando os resultados comparativos das métricas referente ao processo de desenvolvimento de um módulo de software utilizando duas metodologias de desenvolvimento que apresentam-se em evidência. Esta aplicação prática foi conduzida em um ambiente real de desenvolvimento com uma equipe de desenvolvedores e testadores, e que o software produzido foi disponibilizado para o mercado. O estudo serve também de subsídio para a decisão de escolha entre as metodologias para ambiente e parâmetros similares aos do caso estudado.

1.4 Objetivos

O objetivo principal desta dissertação é comparar a metodologia de desenvolvimento ágil, XP – Extreme Programming com uma metodologia do grupo tradicional, RUP – Rational Unified Process e identificar qual delas produzirá um software de maior eficácia em relação à qualidade e o esforço de tempo necessário, dentro de um determinado escopo restrito para a pesquisa.

Os objetivos específicos seguem abaixo relacionados:

- Identificar o tempo necessário para o desenvolvimento e testes, utilizando cada uma das metodologias;
- Descobrir em qual das etapas do ciclo de desenvolvimento ocorreu o maior número de erros;
- Medir a quantidade e tipos de erros produzidos no desenvolvimento em cada uma das metodologias;
- Identificar qual das metodologias obteve o melhor custo benefício, considerando o aspecto erros encontrados versus tempo, dentro do escopo delimitado.

1.5 Limitações do Trabalho

A aplicação prática das metodologias foi feita em um projeto relativamente pequeno, composto por uma equipe de oito pessoas, o mesmo experimento e um projeto de maiores proporções poderia apresentar resultados diferenciados.

As práticas e técnicas recomendadas pelas metodologias foram aplicadas de forma individualizada em dois grupos distintos, não foram testados os resultados de uma possível mesclagem destas metodologias.

Os resultados obtidos foram válidos apenas para o caso delimitado, servindo de parâmetro para projetos com escopos similares, mas não servindo de parâmetro confiável para generalizações acerca da comparação entre as metodologias.

1.6 Estrutura do Trabalho

Este trabalho está organizado em 5 capítulos. O que seguem estão estruturados da seguinte maneira:

O Capítulo II apresenta a fundamentação teórica utilizada neste trabalho. Neste é feito uma revisão do contexto histórico de software, engenharia de software e modelos de desenvolvimento. Também são abordadas as metodologias de Extreme Programming e RUP, foco principal desta pesquisa.

No Capítulo III tem-se o desenvolvimento do trabalho com a descrição do experimento controlado.

No Capítulo IV apresentam-se a análise e os resultados obtidos.

No Capítulo V são descritas as conclusões finais e recomendações.

CAPÍTULO II - FUNDAMENTAÇÃO TEÓRICA

2.1 Software e Engenharia de Software

Durante as décadas de 60,70 e 80 a preocupação e os desafios eram em aumentar o poder de processamento e armazenagem do hardware, barateando também os seus custos. A partir da década de 90 mudou-se o foco das atenções para a elevação da qualidade e redução dos custos de um outro componente de sistemas baseado em computador: o software.

2.1.1 Evolução Histórica do Software

O panorama em que o software foi desenvolvido está diretamente ligado à evolução dos sistemas computadorizados, mais especificamente do hardware. Junto com a mudança dos processadores da válvula para os dispositivos microeletrônicos sugiram softwares mais complexos e sofisticados.

OSBORNE(1979) descreveu este período como uma nova revolução industrial. FEIGENBAUM E MCCORDUCK(1983) pregaram que a informação e o conhecimento controlados por computador seriam o centro principal de poder no século XXI.

Segundo PRESSMAN(1995) durante os primeiros anos de desenvolvimento de sistemas computadorizados o software ficava para segundo plano e a programação era vista como uma arte secundária. O desenvolvimento do software

era feito virtualmente sem nenhuma administração, era projetado sob medida e com uma distribuição limitada. Normalmente era usado pela própria pessoa ou organização que o desenvolvia. Devido a este ambiente personalizado, o projeto era um processo implícito realizado no cérebro de alguém, a documentação muitas vezes não existia e sabia-se muito pouco sobre engenharia de sistemas de computador.

A segunda era foi caracterizada pelo uso e comercialização do produto de software, surgimento das *software houses* e distribuição em massa. Com o crescimento de sistemas baseados em computador começaram ser produzidos milhares de instruções de programa. Todos estes programas e instruções precisavam ser corrigidos quando detectadas falhas ou adaptações conforme as necessidades do usuário. Estas atividades foram chamadas de **manutenção de software** e começaram a consumir quantidade de recursos alarmantes.

A terceira era da evolução começou em meados de 1970. Sistemas distribuídos, redes globais e locais, necessidade de acesso instantâneo a dados exigem muito dos desenvolvedores de software. Caracterizada também pelo uso generalizado de microprocessadores, computadores pessoais, carros, fornos microondas, a terceira era teve o hardware considerado como produto primário e o crescimento da comercialização do software. Neste panorama, a necessidade de desenvolvimento de softwares mais robustos, com mínimos erros e que realmente atendessem aos requisitos que se propunham aumentavam ainda mais.

A partir do momento em que o software atingiu seu grau de importância maior que o do hardware, iniciou-se uma busca constante para o aprimoramento das técnicas de desenvolvimento. Isso sem dúvida se fez necessário, pois com o desenvolvimento de softwares para fins comerciais, as equipes de produção ficaram maiores e as antigas técnicas não mais eram suficientes para dar eficiência aos sistemas.

Neste aspecto, PRESSMAN(1995) afirma que não existe uma abordagem em particular que seja a melhor para a solução dos problemas de software. Entretanto, ao combinar-se métodos abrangentes para todas as fases do desenvolvimento, melhores ferramentas para automatizar estes métodos, blocos de construção mais poderosos para implementação do software, melhores técnicas para a garantia da qualidade e uma filosofia de coordenação predominante, controle e administração, pode-se então conseguir uma disciplina para o desenvolvimento de software, - a qual objetiva contribuir para minimizar problemas em ambientes de desenvolvimento - chamada de Engenharia de Software.

2.1.2 Engenharia de Software

Ainda que muitas definições abrangentes tenham sido propostas, todas ou a grande maioria delas reforçam a exigência da disciplina de engenharia no desenvolvimento do software.

Para SOMMERVILLE(1995), engenharia de software é uma disciplina de engenharia que se preocupa com todos os aspectos da produção de software,

desde as fases precoces de especificação de sistema até para manter o sistema depois que o mesmo entrar em uso.

Também baseando-se na engenharia tem-se a definição de FRITZ BAUER (*apud* PRESSMAN,1995) a qual afirma que: “Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais”.

Em paralelo ao conceito de engenharia, muitas correntes de pensamento também unem uma visão mais gerencial da Engenharia de Software, como pode-se observar na definição de MAFFEO(1992):

> *“Engenharia de software é a área interdisciplinar que engloba vertente tecnológica e gerencial visando a abordar, de modo sistemático, os processos de construção, implantação e manutenção de produtos de software com qualidade assegurada por construção, segundo cronogramas e custos previamente definidos.”*

Para PARNAS (*apud* GHEZZI, JAZAYERI e MANDRIOLLI,1991) a engenharia de software é uma construção de muitas pessoas de muitas versões do software. Esta definição destaca a diferença entre programação e engenharia de software, a programação por ser uma atividade pessoal enquanto a outra respectivamente é essencialmente uma atividade de equipe. Um programador pode até escrever um programa completo enquanto um engenheiro escreve um componente que será utilizado junto com outros componentes desenvolvidos por outros engenheiros.

Em síntese engenharia de software diferencia-se da programação tradicional, ao passo que busca minimizar a distância intelectual do problema a ser resolvido da sua solução computadorizada, otimizando os processos de desenvolvimento de software. (DIJKSTRA *apud* MAFFEO ,1992)

2.1.3 Processos de Desenvolvimento de Software

O processo de desenvolvimento de software ou ciclo de vida de um software, é a execução de um conjunto de atividades associadas, cujo resultado final é um produto de software. Estas atividades normalmente são realizadas por um engenheiro de software. São quatro as atividades principais em todos os processos de software, sendo elas (SOMMERVILLE,1995):

a) Especificação de software (*Software Specification*): As funcionalidades e as restrições de operações devem ser definidas;
b) Desenvolvimento de software (*Software development*): As especificações reunidas do software devem ser produzidas;
c) Validação do software (*Software validation*): O software deve ser testado e assegurado que é exatamente o que o cliente quer;
d) Evolução do software (Software evolution): O software deve reunir as mudanças que o cliente irá precisar.

O ciclo de vida do software nasceu com a finalidade de representar em alto nível o conjunto de atividades envolvidas no desenvolvimento de software. Considerando este processo como um sistema do mundo real o ciclo de vida do software estabeleceria o seu modelo de implementação. (MAFFEO,1992)

Diferentes processos de software organizam as atividades de desenvolvimento de diferentes formas e em diferentes níveis de detalhe. Dependendo da organização podem ser usados processos distintos para o mesmo tipo de produto, no entanto alguns processos são mais eficazes para alguns tipos de aplicação. Se um processo de desenvolvimento inadequado for usado provavelmente reduzirá a qualidade ou as funcionalidades do produto de software. (SOMMERVILLE,1995).

Com o objetivo de simplificar a descrição dos processos de software são usados os modelos de processo de software que os apresentam sobre uma ótica em particular.

2.1.3 Modelos de Processos de Desenvolvimento de Software

O mundo real é complexo e dinâmico, formado por inúmeros componentes que se relacionam entre si. Devido a complexidade deste ambiente são usados modelos no lugar de sistemas reais. O modelo é uma abstração ou aproximação, que é utilizado para simular a realidade (STAIR,1998).

Um modelo por natureza é uma simplificação, assim sendo o modelo de processos de desenvolvimento e software é uma abstração do processo atual que esta sendo descrito (SOMMERVILLE,1995).

Segundo MAFFEO(1992), um dos problemas que tem causado polêmica e trazido desagregação na comunidade de engenheiros de software, é o fato que

muitas vezes o nível de abstração é elevado demais, carecendo mais detalhamento para o entendimento necessário.

Alguns exemplos de tipos de modelos de processos de software que podem ser produzidos, segundo SOMMERVILLE(1995) são:

a) Modelo de fluxo de trabalho (*Workflow model*): Este permite mostrar a seqüência das atividades com suas entradas, saídas e dependências;
b) Modelo de fluxo de dados (*Dataflow model*): Representa o processo como um jogo de atividades, dos quais leva a alguma transformação dos dados;
c) Modelo de papéis/ação (*Role/action model*): Este tipo de modelo representa os papéis das pessoas envolvidas no processo de software e as atividades das quais elas são responsáveis.

A engenharia de software contempla um conjunto de etapas que envolvem métodos, ferramentas e os procedimentos. Estas etapas normalmente são mencionadas como paradigmas de engenharia de software. Este paradigma é selecionado tendo como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usados, os controles e produtos que precisam ser entregues. PRESSMAN(1995)

Uma série de diferentes modelos de ciclos de vida de software foram propostos, cada um exibindo potencialidades e fragilidades, mas todos tendo uma série de fases genéricas em comum. Estes modelos proporcionam uma definição de

alto nível das fases que ocorrem durante o desenvolvimento de software. Dentre eles, os mais conhecidos são: *the waterfall model* (Modelo Cascata ou ciclo de vida clássico), *evolutionary prototyping model* (modelo de prototipação evolucionária), spiral model (modelo espiral), *reusable software model* (modelo de software reutilizável), e *automated software synthesis* (síntese de software automatizada). (Guide to the software engineering body of knowledge,2001).

2.1.4.1 The Waterfall Model *(Modelo Cascata ou Ciclo de Vida Clássico)*

O paradigma *Waterfall* ou ciclo de vida clássico é caracterizado por uma abordagem sistemática, etapas seqüenciais que se inicia ao nível do sistema e progride ao longo da análise, projeto, codificação , testes e manutenção. Modelado em função do ciclo de vida tradicional esta abordagem abrange as seguintes atividades: análise e engenharia de sistemas; análise de requisitos de software; projeto; codificação; testes e manutenção (PRESSMAN,1995):

a) Análise e engenharia de sistemas - Considerando que o software sempre faz parte de um sistema mais amplo, o processo começa com o levantamento de requisitos para o sistema como um todo, e continua com a atribuição de um subconjunto destes requisitos ao software.
b) Análise de requisitos de software - O processo de coleta de requisitos ou levantamento das necessidades e desejos do usuário, se faz necessário para o analista ou engenheiro de software entender a natureza dos programas a serem desenvolvidos assim como as

funcionalidades, desempenho e interface exigida. Os requisitos tanto de sistemas quanto de software são revisados com o cliente servindo também como um mecanismo de aceite de testes.

c) Projeto - O projeto de software é composto por 4 passos distintos: estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização de interface.

d) Codificação - Fase de implementação dos componentes de software definidos e detalhados nas fases anteriores. São trabalhados os aspectos relativos à construção dos programas ou seja traduzir o projeto numa forma legível por máquina.

e) Testes - Fase de aplicação dos planos de teste elaborados em fases anteriores. O processo de teste concentra-se nos aspectos lógicos internos do software e também se todas as suas funcionalidades externas estejam funcionando.

f) Manutenção – Indiscutivelmente o software necessitará de alterações depois que foi entregue ao cliente, isto devido a correções de erros encontrados ou por necessidades funcionais ou de desempenho. A manutenção de software reaplica cada uma das etapas precedentes do ciclo de vida.

Apesar de algumas deficiências o paradigma do ciclo de vida clássico tem a sua importância dentro da engenharia de software e continua sendo o modelo procedimental mais usado. Isto também em virtude das etapas do ciclo de vida clássico serem muito semelhantes ás etapas genéricas que também são aplicados aos demais paradigmas. (PRESSMAN,1995; SOMMERVILLE,1995)

2.1.4.2 *Evolutionary Prototyping Model (Modelo de Prototipação Evolucionária)*

O modelo de prototipação pode ser considerado uma abordagem interessante para os casos em que o cliente define os objetivos gerais para o software, mas não identifica detalhadamente os requisitos de entrada, saída e processamento. Também quando o desenvolvedor precisa certificar-se da eficiência de um algoritmo ou da forma que a interação homem-máquina deva assumir.

Assim como todas as abordagens de desenvolvimento, a prototipação inicia-se com o levantamento de requisitos. Após o cliente e o desenvolvedor reunir-se para definir os objetivos globais do software e quais são os pontos que precisaram ser mais bem detalhados, acontece a construção de um projeto rápido. O projeto rápido conduz a construção de um protótipo que avaliado pelo cliente para refinar os requisitos e servindo também para o desenvolvedor entender melhor o que precisa ser feito. (PRESSMAN,1995)

2.1.4.3 *Automated Software Synthesis (Síntese de Software Automatizada)*

O ciclo de vida pode ser também dirigido por ferramenta**,** que para alguns autores como PRESSMAN(1995), também se utilizam do termo "*técnicas de quarta geração*". O desenvolvedor se utilizará de um conjunto de ferramentas para especificar os requisitos de software num nível elevado e a ferramenta se encarrega de gerar os códigos-fontes.

As técnicas de quarta geração (4GT) estão se tornando parte importante do processo de desenvolvimento de software e provavelmente se tornarão amplamente utilizadas em aplicações de engenharia e tempo real. Com a demanda por software em ascensão, os métodos e paradigmas convencionais provavelmente contribuirão cada vez menos para o todo dos softwares produzidos.

2.1.4.4 *Spiral Model (Modelo Espiral)*

O modelo de ciclo de vida **Espiral** foi desenvolvido para contemplar as melhores características do ciclo de vida clássico quanto o da prototipação. O produto é desenvolvido em uma série de iterações. Cada nova iteração corresponde a uma volta na espiral, e versões mais completas do software são construídas.

O modelo representado pela espiral define quatro importantes atividades: planejamento, análise dos riscos, engenharia e avaliação feita pelo cliente.

Segundo PRESSMAN(1995) o paradigma de modelo espiral é para a engenharia de software a abordagem mais realística para a o desenvolvimento de software em grande escala.

Já para FOWLER(2000) nos últimos anos um novo grupo de metodologias tem surgido para reagir as metodologias tradicionais ou também chamadas de *heavyweight methodologies* (metodologias peso-pesado). As metodologias ágeis têm o compromisso com a não burocracia e serão melhores descritas a seguir.

2.1.5 Metodologias Ágeis

Em fevereiro de 2001 reuniram-se em Utah, EUA, representantes das metodologias *Extreme Programming*, *SCRUM*, *DSDM*, *Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming*, e outros simpatizantes, que até então eram chamadas de metodologias de "peso leve". O objetivo destas pessoas era buscar caminhos alternativos para os processos tradicionais das metodologias "peso pesado" (HIGHSMITH,2001).

O resultado deste encontro foi chamado de ***"Manifesto for Agile Software Development"***, cujo propósito era "Descobrir maneiras de desenvolver softwares fazendo e ajudando outros a fazer". Os valores enfatizados no manifesto foram:

a. indivíduos e interações são mais importantes do que processos e ferramentas;
b. software funcionando é mais importante do que documentação compreensiva;
c. colaboração do cliente é mais importante do que a negociação do contrato;
d. reagir a mudanças é mais importante do que seguir um plano;

Ainda de acordo com HIGHSMITH(2001), o movimento ágil não é uma anti-metodologia, mas sim tem o objetivo de restaurar a credibilidade da palavra metodologia, estabelecendo um equilíbrio. Adotar a modelagem, mas não apenas para arquivar diagramas em um repositório empoeirado; utilizar-se de documentação mas não para desperdiçar centenas de páginas de papel que nunca são lembradas

e raramente atualizadas. Planejar, mas também reconhecer os limites do planejamento em um ambiente turbulento.

Os princípios do movimento ágil são:

a. a maior prioridade é satisfazer o cliente através da entrega precoce e contínua de software de valor;
b. requisitos mutantes são bem-vindos, mesmo quando tarde no desenvolvimento, processos ágeis utilizam as mudanças como uma vantagem competitiva com o cliente;
c. entregar software em funcionamento com freqüência, desde algumas semanas até alguns meses, com preferência para as escalas menores de tempo;
d. pessoas de negócios e desenvolvedores trabalham juntos diariamente do começo ao fim do projeto;
e. implementar projetos em torno de indivíduos motivados, dando a eles o ambiente e o suporte que eles necessitam, confiando neles para o trabalho bem feito;
f. método mais eficaz e eficiente de transmitir informações para a equipe e dentro da equipe é através de conversação face-a-face ;
g. software em funcionamento é a primeira medida de progresso;
h. processos ágeis promovem um desenvolvimento sustentável, patrocinadores, desenvolvedores e usuários devem estar aptos a manter o passo constante indefinidamente;
i. atenção contínua à excelência técnica e a um bom projeto aumentam a agilidade;

j. simplicidade – a arte de maximizar a quantidade de trabalho não feito – é essencial;
k. as melhores arquiteturas, requerimentos e projetos surgem de equipes que se organizam por si só;
l. em intervalos regulares, a equipe reflete sobre como se tornar mais eficiente, então afina e ajusta seu comportamento de acordo.

Uma das metodologias que tem ganhado destaque nos últimos tempos procurando abordar todos estes valores e princípios, é *Extreme Programming* que será analisada a seguir.

2.2 Metodologia eXtreme Programming (XP)

Extreme Programming (XP) é uma metodologia que se encaixa no grupo das metodologias ágeis para desenvolvimento de software. A seguir será apresentada uma introdução sobre suas características, criação e principais práticas.

2.2.1 Introdução

Extreme Programming usa times integrados de programadores, clientes, e gerentes para desenvolver software de alta qualidade em velocidade alta. Reúne

também um conjunto de práticas de desenvolvimento de software já testadas, que estimuladas a sinergia entre elas gerarão vastas melhorias em produtividade global e satisfação do cliente. (HAYES,2001)

Para BECK(2000), criador da *Extreme Programming*, XP é uma metodologia ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança.

Extreme Programming segundo JEFRIES(2002) é uma disciplina de desenvolvimento de software baseada nos valores simplicidade, comunicação, realimentação, e coragem. Trabalha reunindo o time inteiro na presença de práticas simples, através do feedback permite a equipe ver onde ela está e afinar as práticas para situações únicas.

De acordo com FOWLER(2000) em seu artigo, *The New Methodology*, de todas as metodologias peso-leve *Extreme Programming* é que tem tido a maior atenção. Em parte isto é devido ao fato de que os líderes de XP possuem uma habilidade notável, atraindo uma atenção em particular. E também pela capacidade de Kent Beck em atrair o interesse das pessoas pelo assunto.

As raízes da XP estão na comunidade *Smalltalk* com a colaboração íntima de Kent Beck e Ward Cunningham, no final da década de 1980. No início dos anos 90 os dois refinaram suas práticas em numerosos projetos, estendendo o desenvolvimento de software adaptável e orientado a pessoas. O passo mais importante, da prática informal para uma metodologia aconteceu na primavera de

1996. Kent foi convidado para revisar o andamento de um projeto de folha de pagamento para a Chrysler, este estava sendo desenvolvido em *Smalltalk* por uma empresa contratada e estava em dificuldades. Devido à baixa qualidade da base de código, Kent sugeriu que fosse jogado todo fora, iniciando a partir daí o projeto Chrysler C3 que serviu como uma base de aplicação e treinamento para a XP (FOWLER, 2000).

2.2.2 Valores de eXtreme Programming

A XP é uma disciplina de desenvolvimento de software baseada nos seguintes valores: comunicação, simplicidade, *feedback* e coragem. Ela também prega 12 práticas que projetos de XP devem seguir. Muitas destas práticas são antigas, experimentadas e testadas. Ainda que esquecidas por muitos, são incluídas na maioria dos processos planejados. Com base nestas técnicas, XP tece uma sinergia entre elas onde cada uma é reforçada pela outra. (FOWLER, 2000).

A seguir descreve-se detalhadamente os valores da metodologia XP.

2.2.2.1 *Comunicação*

Uma equipe de XP prospera ao entender e compartilhar o problema e o software. O método mais eficiente e efetivo de alcançar compartilhamento é tendo uma comunicação face-a-face. Qualquer coisa que obstrua a comunicação eficiente precisa ser removida (HAYES,2001).

Segundo RIEHLE(2000), o valor de comunicação representa a convicção da metodologia XP de que a chave para um projeto próspero é a comunicação entre os membros do projeto (e consequentemente precisa de ser apoiado através de práticas). Não é declarado por que comunicação é tão importante, mas é seguro assumir que a comunicação é vista como um valor fundamental para a coordenação e colaboração em um projeto.

A XP procura manter a comunicação fluindo através da utilização de diversas práticas que não podem ser feitas sem comunicação. Estas práticas fazem sentido em curto prazo, como testes de unidades, programação em pares e estimativas de tarefas. Como conseqüência destas práticas é que programadores, gerentes e clientes precisam se comunicar (BECK,2000).

É importante lembrar, que em se tratando do valor comunicação em XP, sempre significa comunicação verbal (BECK, 2000).

2.2.2.2 Simplicidade

A simplicidade visa o design do software ser simplificado continuamente, não adicionando partes desnecessárias ao código, mantê-lo claro com nomes significativos e algoritmos sempre quebrados em partes menores, que ele não possua código nem funcionalidade duplicada. A flexibilidade adicionada logo no começo do design pode ser desnecessária e tornar o software complexo, o melhor estado do software no momento da mudança é o simples (BECK, 2000).

Para HAYES(2001), simplicidade é a arte de não maximizar a quantia de trabalho acabado. Jarrete de Dee, antigo CEO da Visa International, diz: "propósito simples, claro e princípios dão lugar ao complexo, comportamento inteligente. Regras complexas e regulamentos dão lugar ao simples, comportamento estúpido".

O valor de simplicidade representa a convicção de XP de que não se deve investir no futuro, mas só nas necessidades imediatas do projeto. Estando por baixo deste valor é a suposição que o futuro não pode ser predito confiantemente e preocupar-se com isso hoje não é economicamente inteligente (RIEHLE, 2000).

Complementando, o valor simplicidade prega que se mantenha o sistema tão simples quanto possível. Isso significa não gastar tempo e esforço para implementar características que podem não ser necessárias depois. Projetos de XP fazem a coisa mais simples possível, confiante que pode ser mudado depois a pequeno custo adicionado (HAYES, 2001).

Simplicidade e comunicação possuem uma relação muito boa de apoio mútuo. Quanto mais contínua é a comunicação, mas evidente fica o que deve e o que não deve ser feito.

A simplicidade deve ser aplicada em todas as atividades do processo, procurando sempre a coisa mais simples que possivelmente funcione.

2.2.2.3 Feedback

O valor *feedback* significa que é importante ter um sistema rodando a qualquer hora. Isso dá para o desenvolvedor a informação fidedigna sobre seu funcionamento (aqui o *feedback* não está direcionado aos seres humanos mas, sim a respeito do estado de desenvolvimento). Efetivamente, o sistema e seu código servem como o oráculo incorruptível para informar sobre o progresso e estado do desenvolvimento. O *feedback* também é um meio para orientação e tomada de decisão, para onde ir (RIEHLE, 2000).

O posicionamento concreto do estado corrente do projeto é absolutamente essencial, fazendo que todo o problema seja evidenciado o mais cedo possível para que possa ser corrigido o quanto antes, da mesma forma fazendo com que as oportunidades sejam descobertas o mais precocemente possível para que possam ser mais bem aproveitadas.

O feedback funciona em conjunto com comunicação e simplicidade. Quanto mais feedback existir, mais fácil será a comunicação. Se os indivíduos envolvidos no projeto estiverem se comunicando claramente, aprenderão sobre o sistema e saberão o que testar.

2.2.2.4 Coragem

Equipes de desenvolvimento de software prósperas, precisam constantemente trabalhar na extremidade do caos, eles precisam agir rapidamente sem perder o controle. Isto às vezes significa cometer algumas falhas mas sem

perder a coragem para se tomar as decisões certas mesmo quando se pressionado para fazer outra coisa (HAYES, 2001).

Na verdade é preciso que a equipe não tenha medo de: parar quando se está cansada; deixar as decisões do negócio para o cliente; apontar um problema no projeto; pedir ajuda ao parceiro e aos clientes quando necessário; simplificar o código que já está funcionando; dizer ao cliente que não será possível implementar um requisito no prazo estimado; fazer alterações no processo de desenvolvimento, ou muitas vezes jogar o código fora e recomeçar. Para tudo isto é preciso muita coragem (LONGMAN,1998).

Os valores de comunicação, simplicidade, feedback e coragem servem como suporte para doze práticas da XP, que serão apresentadas no tópico seguinte.

2.2.3 Práticas de eXtreme Programming

As doze práticas promovidas pela XP se forem examinadas individualmente apresentarão falhas, mas uma das forças da XP é que as práticas se combinam de um modo mútuo apoiando-se. Juntas as práticas conduzem a um complexo, comportamento emergente. Cada prática tem sua função para manter o custo de mudança baixo (HAYES,2001).

De acordo com BECK(2000) as doze práticas da XP são mapeadas em três categorias, a primeira englobando as práticas de programação, a segunda as práticas orientadas para a equipe e a terceira contempla os processos através dos

quais a equipe de programação relaciona-se com o cliente. As práticas estão dividas nas categorias da seguinte forma:

a) *Programming – Simple design* (projeto simples), *testing* (testes), *refactoring* (reconstrução), *coding standards* (código padrão);
b) *Team - Collective ownership* (código coletivo), *continuous integration* (integração continua), *metaphor* (metáfora), *coding standards* (código padrão), 40-*hour week* (40 horas por semana), *pair programming* (programação em par), *small releases* (versões pequenas);
c) *Process - On-site customer* (cliente no local), *testing* (testes), *small releases* (versões pequenas), *planning game* (planejamento do jogo).

2.2.3.1 Simple Design

Manter o custo de manutenção baixo significa manter o design tão simples quanto possível. Também significa não implementar *features* que podem ou não serem utilizadas mais tarde. Em um projeto XP se faz as coisas tão simples quanto possível, acreditando que elas poderão ser alteradas mais tarde com um pequeno custo adicional (HAYES,2001).

2.2.3.2 Testing

Todo o requerimento é refletido em um teste de aceitação. Os testes de aceitação são produzidos pelos próprios clientes. Os programadores escrevem os casos de teste antes de escreverem o código, usam o teste para focar o que tem que ser alcançado e também para especificar a interface. Todos os testes são automatizados e executados regularmente (HAYES,2001).

2.2.3.3 Refactoring

É o processo de melhorar o *desing* do código sem afetar seu comportamento externo. O *refactoring* é feito de forma que o código seja mantido tão simples quanto possível, pronto para qualquer mudança futura (WAKE,2000).

2.2.3.4 Coding Standards

Codificar é uma atividade de equipe. Com o passar do tempo pessoas diferentes trabalharão em diferentes pedaços do código, e disparidades de estilo de codificação e convenções fazem o código mais difícil de se trabalhar. Para a equipe ser efetiva precisa-se de um código padrão, do qual ao ser visto parece ser escrito pelo mesma pessoa (HAYES,2001).

2.2.3.5 Collective Ownership

Qualquer pessoa da equipe pode alterar qualquer código, contanto que apoiado por um parceiro, obedecendo aos padrões e assegurado por todo o trabalho de testes quando eles terminarem. O benefício disto é que remove os gargalos e os problemas de distorção de arquitetura que podem surgir (HAYES,2001).

2.2.3.6 Continuous Integration

A equipe XP trabalha em etapas curtas e integra o código periodicamente. Isto significa que os problemas de integração são descobertos logo ao serem criados, desta forma é razoavelmente fácil retificá-los. Integração contínua evita

desenvolvimentos incompatíveis e ajuda assegurar que toda a equipe esteja trabalhando na mais recente versão do sistema todo o tempo (HAYES,2001).

2.2.3.7 Metaphor

A metáfora de sistema dá para a equipe um vocabulário consistente para discutir os problemas e suas soluções. Significa falar sobre o sistema em termos de objetos de negócio (HAYES,2001).

2.2.3.8 40-Hour Week

Desenvolvimento de software é um exercício criativo, e ninguém pode ser criativo se está exausto. Restringindo o número de horas por uma semana de trabalho mantém as pessoas descansadas, reduz a sobrecarga e melhora a qualidade do produto acabado. (HAYES,2001)

2.2.3.9 Pair Programming

Toda a produção de desenvolvimento é feita por duas pessoas que compartilham uma máquina. Isto é muito mais eficiente que ter duas pessoas programando separadamente. Os pares freqüentemente giram, assim o conhecimento e a experiência circulam pela equipe e também o código é revisado continuamente (HAYES,2001).

2.2.3.10 Small Releases

As *releases* devem ser tão pequenas quanto possível, entregando bastante valor ao negócio. Equipes de XP podem executar lançamentos em ciclos de algumas semanas, porque estão trabalhando em passos pequenos, têm testes para efetuar uma regressão, e estão integrando continuamente. Alguns projetos de XP executam um lançamento diariamente. (HAYES,2001)

2.2.3.11 On-Site Customer

Nenhuma exigência escrita está completa e sem ambigüidade. Os programadores sempre precisam ter acesso a um cliente para esclarecer os requisitos, não importando a quantidade de esforço dedicada na especificação original. Uma equipe de XP mantém isto simples saltando muito o esforço destinado para a especificação, e tendo um cliente todo o tempo disponível para os programadores. Programadores de XP não adivinham os detalhes de uma *feature*, ao invés eles perguntam para o cliente (HAYES,2001).

2.2.3.12 Planning Game

Classifica as negociações regulares em cima das funcionalidades que acontecem em um projeto de software e as transforma em um jogo. O Jogo de Planejamento é realizado em repetição para determinar que funcionalidade irá no próximo lançamento. Programadores tomam decisões técnicas (estimativas) e os clientes tomam decisões empresariais, selecionando as funcionalidades com a maioria de benefícios (HAYES,2001).

2.3 Metodologia Rational Unified Process (RUP)

A seguir será apresentada uma introdução sobre RUP, assim como suas práticas e conceitos.

2.3.1 Introdução

No decorrer da década de 90, o paradigma da orientação objeto ganhou força tornando-se mais evidente suas vantagens dentro da engenharia de software. Diante deste cenário começaram a proliferar inúmeras metodologias baseadas neste conceito, revivendo as experiências negativas que já haviam ocorrido com as metodologias estruturadas. Devido a este motivo ganhou força à idéia de uma metodologia de desenvolvimento de software unificada, que se aproveitasse da experiência e conceitos da linguagem UML (Unified Model Language, Linguagem de Modelagem Unificada). Dentre as metodologias baseadas neste paradigma está o RUP (*Rational Unified Process*) (SILVA e VIDEIRA,2001, BOOCH 2000).

RUP surgiu há pouco tempo, mais precisamente em 1998, apesar disso contempla em suas origens idéias e experiências vividas nos últimos trinta anos, em especial abordagens seguidas na Ericson onde trabalhou Ivar Jacobson até 1987. Na seqüência Jacobson fundou a empresa Objectory AB, a qual foi adquirida pela Rational em 1995. A Rational por sua vez vinha desenvolvendo desde 1981 um conjunto de iniciativas com o objetivo de criar um ambiente interativo que permitisse aumentar a produtividade do desenvolvimento. Tinha como um dos seus principais mentores Philippe Kruchten, mas foi com a entrada de Grady Booch, James

Rumbaugh e Ivar Jacobson, denominados de “três amigos”, que começaram as iniciativas para unificação das metodologias, cujo resultado foi o *Rational Objectory Process* que a partir de 1998 passou-se chamar *Rational Unified Process.*

O *Rational Unified Process* é um processo de engenharia de software, que procura disciplinar a atribuições de tarefas e responsabilidades dentro de uma estrutura de desenvolvimento de software. Sua meta principal é garantir a produção de software com alta qualidade satisfazendo as necessidades dos seus usuários, dentro de um cronograma e orçamento previsível (RATIONAL SOFTWARE CORPORATION,2002).

O RUP também aumenta a produtividade da equipe provendo fácil acesso a bases de conhecimento com diretrizes e modelos para atividades de desenvolvimento críticas, proporcionando que todos os membros do desenvolvimento tenham acesso a mesma base de conhecimento. Desta forma assegura-se que todos na equipe compartilhem processos e um idioma comum, tendo assim a mesma visão do software a ser desenvolvido. (RATIONAL SOFTWARE CORPORATION,2002)

Uma característica das atividades desta metodologia é de criar e manter modelos, em vez de focar a produção de grande quantidade de documentos, procura enfatizar o desenvolvimento e manutenção de modelos ricos na representação semântica.

O RUP é um processo configurável pois um único processo não é satisfatório para todo o desenvolvimento de software. O processo unificado ajusta-se tanto para pequenas equipes de desenvolvimento quanto para grandes organizações. O Processo Unificado é fundado em uma arquitetura de processo simples e claro, e ainda pode ser adaptado para acomodar situações diferentes, dependendo da necessidade da organização Possui várias práticas que integradas completam sua estrutura de atuação, que serão vistas a seguir.

2.3.2 As Práticas do RUP

O *Rational Unified Process*, reúne muitas das melhores práticas em desenvolvimento de software moderno e coloca a disposição dos projetos e organizações. Na seqüência serão detalhadas estas práticas.

2.3.2.1 Develop Software Iteratively

Devido à sofisticação dos sistemas de software de hoje é impossível que se defina o problema por inteiro, projete a solução inteira, construa o software e o teste até o produto final. As metodologias unificadas sugerem uma aproximação iterativa e incremental, permitindo uma compreensão crescente do problema por refinamentos sucessivos, rumando para uma solução efetiva sobre repetições múltiplas. (SILVA e VIDEIRA,2001)

O RUP suporta uma aproximação iterativa para desenvolvimento, que identifica os fatores de alto risco em todas as fases do ciclo de vida, reduzindo significativamente os riscos do projeto. A aproximação iterativa ajuda a atacar os

riscos por demonstrar freqüentemente o progresso, lançando executáveis que permitem o envolvimento do usuário e o seu *feedback*.

Devido cada repetição terminar com o lançamento de um executável a equipe de desenvolvimento fica focada no resultado, e freqüentemente ajuda checar o status garantindo que o projeto está dentro do cronograma. Uma outra vantagem da aproximação iterativa é tornar mais fácil a adaptação das mudanças táticas em requisitos, características e cronogramas. (RATIONAL SOFTWARE CORPORATION,2002)

2.3.2.2 Manage Requirements

Nesta prática o *Rational Unified Process*, descreve como extrair, organizar e documentar funcionalidades exigidas; facilmente captura e comunica exigências empresariais. As noções de caso de uso e cenários provaram ser um modo excelente para capturar exigências funcionais e assegurar que estas conduzam o design, implementação e testes do software. Assim, torna-se muito mais provável que o sistema final cumpra as necessidades do usuário. (RATIONAL SOFTWARE CORPORATION,2002)

2.3.2.3 Use Component-Based Architectures

O processo enfoca o desenvolvimento baseando-se em uma arquitetura executável robusta, antes de se comprometer recursos para o desenvolvimento completo. Descreve também como projetar uma arquitetura elástica e flexível, que

acomode mudanças e seja intuitivamente compreensível, promovendo mais efetivamente a reutilização de software.

O *Rational Unified Process* apóia o desenvolvimento de software baseado em componentes. Componentes são módulos não triviais, são subsistemas que cumprem uma função clara. O RUP provê uma aproximação sistemática para definição de uma arquitetura que use componentes novos e já existentes. (RATIONAL SOFTWARE CORPORATION,2002)

2.3.2.4 Visually Model Software

O processo mostra visualmente como o modelo de software captura a estrutura e o comportamento das arquiteturas e componentes. Isto permite esconder os detalhes e escrever código usando blocos gráficos. As abstrações visuais ajudam a comunicar aspectos diferentes do software; ver como os elementos do sistema se ajustam juntos; ter certeza que os blocos são consistentes com seu código; manter consistência entre o design e sua implementação; e promover a comunicação sem ambigüidade. (RATIONAL SOFTWARE CORPORATION,2002)

2.3.2.5 Verify Sotware Quality

Desempenho de aplicação e confiabilidade pobres são fatores comuns que dramaticamente inibem a aceitabilidade das aplicações de software de hoje. Consequentemente, deveria ser revisada a qualidade com respeito às exigências baseadas em confiabilidade, funcionalidade, desempenho de aplicação e desempenho de sistema. O RUP ajuda no planejamento, projeto, implementação,

execução, e avaliação dos testes. A taxa de qualidade é construída durante o processo, em todas as atividades, envolvendo todos os participantes, usando medidas objetivas e criteriosas, e não tratada como uma reflexão tardia ou uma atividade separada executada por um grupo separado. (RATIONAL SOFTWARE CORPORATION,2002)

2.3.2.6 Control Changes To Software

A habilidade para administrar mudanças sugere que se tenha certeza da aceitação da mudança, conseguindo localizá-la quando necessário, pois isso é essencial em um ambiente onde as mudanças são inevitáveis. O processo descreve como controlar, rastrear e monitorar as mudanças, permitindo o sucesso do desenvolvimento da iteração. Também orienta o estabelecimento de espaços de trabalho seguro para cada desenvolvedor, provendo o isolamento das mudanças feitas em outro espaço de trabalho e controlando mudanças de todos os artefatos de software. (RATIONAL SOFTWARE CORPORATION,2002)

A arquitetura do RUP apresenta-se dividida em duas dimensões, as quais refletem as duas visões em que um sistema pode ser descrito: componentes dinâmicos e componentes estáticos, os quais serão descritos na seqüência.

2.3.3 Dimensões dos Processos

Os processos podem ser descritos em duas dimensões: a primeira demonstra o tempo, os aspectos dinâmicos do processo e como ele é ordenado. É

representada em termos de ciclos, fases, interações e marcos (*milestones*). A segunda dimensão representa os aspectos estáticos do processo, descreve os termos das atividades, artefatos, participantes do processo e o fluxo de trabalho (*workflow*). (SILVA e VIDEIRA,2001)

2.3.3.1 Componentes Dinâmicos

O ciclo de vida do software é dividido em ciclos, cada ciclo trabalhando em uma versão nova do produto, pode-se encarar um projeto como uma seqüência de fases, cada uma com várias iterações. De acordo com o RUP esta fases são as seguintes:

a) Concepção (*Inception*) : Durante a fase de concepção, estabelece-se uma visão global para o sistema e delimita-se o âmbito de projeto;
b) Elaboração (*Elaboration*): O propósito desta fase é analisar o domínio do problema, especificar as funcionalidades e desenhar a arquitetura;
c) Construção (*Construction*): Durante a fase de construção, são desenvolvidos os componentes e as características da aplicação são integradas no produto, todas estas características são testadas;
d) Transição (*Transition*): O propósito da fase de transição é disponibilizar o produto de software para a comunidade de usuário, bem como garantir o respectivo sucesso. (RATIONAL SOFTWARE CORPORATION,2002,SILVA e VIDEIRA,2001).

Cada fase do *Rational Unified Process* é concluída com um ponto de controle (*milestone*), um marco que serve para comparar os resultados obtidos com os objetivos principais e também para a tomada de decisões críticas.

2.3.3.2 Componentes Estáticos

O processo descreve quem está fazendo o que, como e quando. O RUP é representado usando quatro elementos para modelagem: *workers, activities, artifacts e workflows.*

Esta visão centrada no conceito de *workflow* está dividida em *workflow* de processamento e *workflow* de suporte. No primeiro grupo apresentam-se os *workflows* de :

a) Modelação de processos de Negócio (*Business Modeling*): Um dos principais problemas com a maioria dos esforços de engenharia empresarial, é que a engenharia de software e a comunidade de engenharia empresarial não se comunicam corretamente. Isto resulta que a produção de engenharia empresarial não é corretamente usada como entrada para o desenvolvimento de software, e vice-versa. Para solucionar este problema o RUP disponibiliza um idioma comum e compreensível para ambas comunidades, que é usado na modelação de processos de negócio para o entendimento dos processos da organização;
b) Requisitos (*Requirements*): O objetivo do *workflow* de requerimentos é descrever o que o sistema deve fazer e permitir a concordância do

programador e do cliente. Para isto é agrupado um conjunto de atividades relacionadas com a identificação e modelagem dos requisitos do sistema;

c) Análise e Desenho (*Analysis and Design*): A meta deste *workflow* é mostrar como a implementação será realizada, tem como resultado o desenho e a análise do modelo, sendo que o desenho pode ser usado como uma abstração do código fonte;
d) Implementação (*Implementation*) : Os propósitos da implementação são: definir a organização do código, implementar classes e objetos, testar os componentes desenvolvidos;
e) Testes (*Test*) : Os objetivos dos testes são verificar a interação entre os objetos; verificar a integração formal entre todos os componentes do software; certificar que todos os requisitos foram corretamente implementados; assegurar que os defeitos foram identificados antes da instalação do software.
f) Instalação (*Deployment*) : Visa disponibilizar o sistema para seus usuários finais, engloba atividades de lançamento externo, distribuição do software, instalação e suporte aos usuários. (RATIONAL SOFTWARE CORPORATION,2002)

O segundo grupo de *workflows*, denominados *workflows* de suporte, reúnem atividades que acontecem no decorrer do projeto e contribuem diretamente para o sucesso do produto final. Estas atividades são a gestão do projeto (*project management*), gestão da configuração e das alterações (*configuration and change*

management) e a definição do ambiente de desenvolvimento (*Enviroment*). (SILVA e VIDEIRA,2001)

Os conceitos do RUP são um tanto quanto inovadores se comparados com as formas mais tradicionais e instituídas de desenvolvimento de software. A principal diferença comparada às abordagens mais clássicas são de que as fases decorrem em várias iterações e se estendem ao longo do tempo por várias fases.

2.4 Linguagem de Modelagem Unificada (UML)

A UML é uma notação com características diagramáticas utilizada para modelagem de sistemas, que se utiliza de conceitos orientados a objetos (LARMAN, 2000).

Esta notação surgiu como uma tentativa de padronizar os artefatos de análise e desing, unificando as notações baseadas em objeto. A UML é uma linguagem usada para especificar, visualizar e documentar os artefatos de um sistema orientado em objeto (QUATRANI,2001).

CAPÍTULO III - METODOLOGIA

No decorrer deste capítulo será apresentada a estrutura metodológica que foi empregada no desenvolvimento da pesquisa. Classifica-se a pesquisa, quanto ao seu tipo, detalha-se quais foram os caminhos percorridos para chegar aos objetivos propostos, apresenta-se todas as especificações técnicas, materiais e recursos utilizados, assim como se expõe a forma de coleta, tabulação e análise dos dados.

3.1 Classificação da Pesquisa

Esta pesquisa pode ser classificada quanto a sua natureza, como pesquisa aplicada pois objetiva gerar conhecimentos para aplicação prática do desenvolvimento e testes de softwares, orientados à solução de determinados problemas. No que se refere à abordagem e objetivos, trata-se de uma pesquisa quantitativa e exploratória respectivamente, pois além da pesquisa bibliográfica, os resultados obtidos com o estudo de caso são quantificados para proporcionar condições de classificá-los e analisá-los.

3.2 Desenvolvimento

Após uma revisão literária sobre o assunto, foram apresentados os objetivos da pesquisa a um grupo de desenvolvedores e testadores de software, da empresa Ciss Automação Comercial, localizada na cidade de Dois Vizinhos, estado do Paraná, os quais foram convidados a participar do desenvolvimento de um software

utilizando as Metodologias XP e RUP. A proposta foi avaliada e aceita pela equipe em junho de 2002.

Esta equipe foi composta por oito pessoas, destes, seis têm formação acadêmica em Ciência da Computação ou Sistemas de Informação, um em Administração e o último em Economia. Foram revisados juntamente com a equipe os conceitos das metodologias, para que todos os envolvidos tivessem o domínio das técnicas a serem utilizadas no projeto.

Alguns membros da equipe foram divididos em dois grupos, o primeiro com a responsabilidade de desenvolver um componente de software utilizando XP e outro desenvolver o mesmo componente utilizando RUP. O restante da equipe realizou tarefas comuns aos dois grupos. Os papéis e as responsabilidades dos membros da equipe, ficaram distribuídos da seguinte forma:

Quadro 1: Papéis e responsabilidades dos membros da equipe

Quantidade de Pessoas	**Papel**	**Responsabilidade**
1	Engenheiro de software (Desenvolvedor)	Analisar, projetar, implementar e testar os componentes de software, utilizando a Metodologia RUP (*Rational Unified Process*).
2	Engenheiro de software (Desenvolvedor)	Analisar, projetar, implementar e testar os componentes de software, utilizando a Metodologia XP(*Extreme Programming*).
1	DBA (Administrador da base de dados)	Prover para os dois grupos de desenvolvedores as definições de tabelas, triggers e o modelo do banco de dados.
1	Analista de Negócios	Fornecer para os dois grupos de desenvolvedores os requisitos do negócio, bem como suas definições.
1	Testador	Elaborar e executar os planos de testes.
1	Pessoa de apoio	Coletar os dados e organizar os artefatos produzidos por cada etapa do desenvolvimento.
1	Líder de projeto	Acompanhar o andamento das atividades, planejar e alocar recursos.

Cada membro da equipe utilizou microcomputador K6 II 800Mhz, com 128MB de memória. As ferramentas de software que deram suporte às etapas do ciclo de desenvolvimento foram as seguintes:

Quadro 2: Ferramentas utilizadas

Atividade	Ferramenta	Fabricante
Documentação Requisitos	Microsoft Office Word	Microsoft
Análise e Projeto	Rational Rose 98	Rational
Modelagem de Banco de dados	ErWin 3.5.2	Platinum
Modelagem de Processos	BpWin	Platinum
Gerência de Configuração	Source Safe 6.0	Microsoft
Desenvolvimento	PowerBuilder 8.0.3	Sybase
Banco de dados	Sybase Anywhere 8	Sybase

Após a equipe estar montada, os papéis e as responsabilidades definidas, infra-estrutura de hardware e software disponível, foram repassados aos engenheiros de software as regras de negócio, modelo de dados e os requisitos primários. É importante ressaltar que ambos os grupos, XP e RUP, partiram deste mesmo ponto em comum, e que o software utilizado como base da pesquisa foi realmente solicitado por um cliente. Ao final do projeto o produto será entregue ao usuário final. Como serão desenvolvidos 2 projetos em paralelo – um em cada metodologia - , o que obtiver o menor custo/beneficio, levando-se em consideração que custo=horas/homem e benefício=qualidade do produto(menor número de erros), será o escolhido para entrega ao cliente.

3.2.1 Modelagem do Negócio

As empresas de uma maneira geral trabalham com a política de comissionamento de suas equipes de vendas, desta forma se faz necessário um controle das operações de débito, crédito e também o saldo destas comissões.

Este módulo a ser desenvolvido terá a finalidade de permitir ao usuário lançar valores de comissão para os vendedores cadastrados no sistema, deverá ser considerado que já existe no sistema o recurso de cadastramento dos vendedores. O relatório completo da modelagem e o detalhamento das especificações está no Anexo I.

Após o recebimento deste relatório as equipes de XP e RUP partiram para próximas etapas do ciclo de desenvolvimento. O ciclo completo do desenvolvimento ficou divido em 2 sub-ciclos ou duas iterações. No primeiro ciclo foram implementados os requisitos pelos desenvolvedores e efetuados os testes de aceitação pela equipe de testes, no segundo ciclo foram implementadas as correções pelos desenvolvedores dos erros detectados pela equipe de testes no primeiro ciclo, assim como os re-testes após as correções terem sido efetuadas. Todas estas atividades serão melhores descritas na seqüência.

3.2.2 Atividades do Grupo de XP

As atividades do grupo de XP foram orientadas de acordo com as práticas recomendadas pela metodologia, as quais se dividem nas categorias de processos, programação e equipe.

3.2.2.1 Processos

Dentre as práticas que envolvem os processos, os desenvolvedores iniciaram o planejamento (*planning game*), fazendo definição incremental dos requisitos do sistema e a criação das *user stories*, (Anexo III) pois na metodologia XP não são definidas as especificações formalmente com relatórios e diagramas e muito menos definidos já na sua totalidade.

As demais práticas que foram seguidas pelos desenvolvedores dentro do tópico de processo foram: a presença constante e a interação com o cliente, representado pelo analista de negócios (*client on-site),* o qual repassou os requisitos e as necessidades do cliente; liberação de pequenas releases para avaliação do mesmo (*small releases*); os processos de testes também foram seguidos, as *user stories* utilizadas como insumos de entrada para criação do plano de teste de aceitação, assim como a criação dos scripts de teste (*Testing*), que podem ser observados no Anexo V.

Uma u*ser story* pode ter um ou mais testes de aceitação, conforme definição do cliente e/ou usuários que estarão desenvolvendo estes testes para assegurar que as funcionalidades do sistema estejam de acordo com as especificações. Estes testes garantem também ao cliente e aos desenvolvedores que o sistema em desenvolvimento está progredindo na direção certa. Enquanto todos estes testes criados (cliente/usuário) não forem satisfeitos o produto não pode ser entregue.

3.2.2.2 Programação

A codificação deste sistema foi feita em dupla, ou seja, dois desenvolvedores na mesma máquina (*Pair Programming*). A padronização do código (*coding standards*) também foi aplicada, pois no revezamento dos desenvolvedores o código escrito por um, deveria ser entendido pelo outro de forma clara sem que nenhum dos dois precisasse justificar a parte desenvolvida.

Antes dos desenvolvedores iniciarem a criação do código definitivo, eles trabalharam na construção dos scripts de testes ou testes unitários. Estes scripts eram alterados e executados cada vez que uma nova implementação era feita no código, para tentar assegurar que não ocorressem impactos negativos, conforme scripts de testes do Anexo V. Assim, eram armazenados num repositório junto ao código do sistema de forma que a execução fosse automatizada, garantindo uma proteção essencial contra erros. Todas as alterações e execuções destes testes foram cronometradas para avaliação entre o tempo de alteração do teste e quantidade de erros encontrados.

Além disso, foi implementada a característica de simplicidade no projeto do sistema (*Simple design*), para que os desenvolvedores codificassem apenas as funcionalidades essenciais ao sistema, assim, gastando menos tempo do que desenvolver um código complexo; também sendo essencial esta simplicidade para as alterações que os desenvolvedores faziam no código.

Durante a codificação também se aplicou a prática de reconstrução de alguns códigos (*Refactoring*), foram eliminadas funcionalidades desnecessárias para o projeto, fazendo com que a simplicidade aumentasse, evitando a desordem e a complexidade. Em resumo os desenvolvedores procuraram não adicionar funcionalidades antes do necessário, apenas as funcionalidades específicas de cada u*ser story*, facilitando a construção e execução dos testes, alteração e reconstrução do código.

3.2.2.3 Equipe

Das práticas relacionadas à equipe, que foram aplicadas as que se destacaram foram a de código coletivo (*Collective ownership*), pois nenhum dos dois desenvolvedores se consideravam pai do código, pois o mesmo tinha sido criado em dupla, possibilitando que qualquer um dos dois executassem a manutenção, e a outra foi a integração contínua (*continuous integration*) que ao final de cada componente desenvolvido no sistema era integrado ao todo e executados todos os *scripts* de testes.

Terminada a etapa de desenvolvimento do software da equipe XP, passou-se para os testes de aceitação. Estes testes foram realizados pelo analista de testes que teve a responsabilidade de criar e executar os planos de testes. Esta etapa será descrita, logo após o tópico das atividades do grupo de RUP.

3.2.3 Atividades do Grupo de RUP

O grupo de RUP trabalhou paralelamente ao grupo XP no ciclo de vida do software, seguindo as fases de concepção (*inception*), elaboração (*elaboration*), construção (construction) transição (transition), cada uma com várias iterações. Dentro destas fases os processos também foram seguidos: os workflows de requisitos (*requirements*), análise e desenho (*analysis and design*), implementação (*implementation*), testes (*test*), na seqüência os mesmos serão melhores descritos.

3.2.3.1 Levantamento e detalhamento dos requisitos

Com base nos requisitos recebidos foi desenvolvido um documento contendo o objetivo geral e o objetivo específico. Nesta fase também foram detalhados os requisitos e especificadas as funções do sistema (funcionalidades do sistema que têm ligação direta com todos os requisitos recebidos), sendo estas detalhadas ao máximo para que na fase de implementação não tivesse nenhuma dúvida.

3.2.3.2 Desenho e arquitetura

Nesta fase definiram-se os atributos do sistema como:

- tempo de resposta para mostrar telas ou fazer pesquisas pelo usuário;
- tolerância a falhas: emitir mensagens confirmando as operações do usuário;
- plataforma(s) utilizada(s) para o sistema e hardware necessário para impressão.

Após o levantamento dos atributos, definiu-se o protótipo de interface do sistema, contendo todos os tipos de objetos utilizados e suas funcionalidades, assim como a imagem de cada uma das telas criadas como protótipo a serem aprovadas pelo cliente.

Foram desenvolvidos os casos de uso do sistema, com a utilização da ferramenta *Rational Rose 98*, a partir dos requisitos estabelecidos no detalhamento.

As tabelas de banco de dados envolvidas já estavam modeladas e disponíveis, bastando ao desenvolvedor utilizá-las para documentação.

Por último foi efetuado nesta fase o detalhamento das especificações técnicas de codificação dos componentes do sistema. Utilizou-se também das tabelas de banco de dados para a documentação técnica referente a utilização dos campos na codificação do sistema. Os artefatos originados nesta fase estão no Anexo II.

3.2.3.3 Implementação

Nesta fase foram implementados todos os componentes e especificações do sistema descritos nas fases anteriores. Portanto, nesta fase o desenvolvedor criava e executava os testes unitários toda vez que um código era escrito e/ou modificado. Os erros eram corrigidos assim que encontrados fazendo com que o desenvolvedor implementasse o sistema de forma com que a função codificada ficasse disponível

no menor tempo possível, e conseqüentemente iria se integrar as outras funções mais rapidamente.

3.2.3.4 Testes

Durante o desenvolvimento do projeto RUP, foram realizados os seguintes testes:

- Teste unitário: este tipo de teste era executado cada vez que uma nova alteração no código era feita.
- Teste de integração: teste realizado para verificar a integração das interfaces implementadas com relação as interfaces já desenvolvidas.
- Teste de sistema: foram testadas as questões de desempenho e comportamento do sistema com relação ao tempo das transações.
- Teste aceitação: testados todos os requisitos e funcionalidades criados no sistema, conforme documentação dos mesmos nas fases.

Durante a execução das atividades do grupo de XP e RUP, foram cronometradas e quantificadas algumas métricas, no tópico seguinte será descrito como foi realizada esta tarefa.

3.2.4 Coleta de Dados

Os grupos de XP e RUP foram acompanhados por uma pessoa de apoio que durante as atividades desenvolvidas efetuava o registro dos erros conforme aconteciam. Estes erros foram categorizados conforme HUMPHREY(1995) e apresentados no Quadro 3.

Quadro 3: Categorias de Erros

Índice dos tipos de erros padrões

10	**Documentação**	10.1 - Comentários
		10.2 - Mensagens
20	**Sintaxe**	20.1 - Escrita
		20.2 - Pontuação
		20.3 - Tipos
		20.4 - Formato das instruções
30	**Pacote**	30.1 - Gerenciamento mudança
		30.2 - Bibliotecas de programas
		30.3 - Controle de versão
40	**Associação**	40.1 - Declaração
		40.2 - Nomes duplicados
		40.3 - Escopo
		40.4 - Limites
50	**Interface**	50.1 - Chamadas de procedimentos e referências
		50.2 - I/O
		50.3 - Formatos
60	**Checagem**	60.1 - Mensagens de erros
		60.2 - Verificações inadequadas
70	**Dados**	70.1 - Estruturas
		70.2 - Conteúdos
80	**Funções**	80.1 - Lógica
		80.2 - Ponteiros
		80.3 - Loops
		80.4 - Recursividade
		80.5 - Computação
		80.6 - Defeitos de funções
90	**Sistema**	90.1 - Configuração
		90.2 - Tempo
		90.3 - Memória
100	**Ambiente**	100.1 - Projeto
		100.2 - Compilação
		100.3 - Teste
		100.4 - Outros problemas de suporte de sistemas

Os erros que foram registrados do grupo de XP, estão apresentados Tabela 1 e os de RUP na Tabela 2.

Tabela 1: Quantidade de erros registrados na fase de desenvolvimento para o grupo XP

Número do Tipo	Nome do tipo	Quantidade de erros					
10	**Documentação**	**10.1**	**10.2**				
Total	0						
20	**Sintaxe**	**20.1**	**20.2**	**20.3**	**20.4**		
Total	11	4		1	6		
30	**Pacote**	**30.1**	**30.2**	**30.3**			
Total	0						
40	**Associação**	**40.1**	**40.2**	**40.3**	**40.4**		
Total	2	2					
50	**Interface**	**50.1**	**50.2**	**50.3**			
Total	0						
60	**Checagem**	**60.1**	**60.2**				
Total	0						
70	**Dados**	**70.1**	**70.2**				
Total	0						
80	**Funções**	**80.1**	**80.2**	**80.3**	**80.4**	**80.5**	**80.6**
Total	8	8					
90	**Sistema**	**90.1**	**90.2**	**90.3**			
Total	0						
100	**Ambiente**	**100.1**	**100.2**	**100.3**	**100.4**		
Total							
TOTAL	21						

Tabela 2: Quantidade de erros registrados na fase de desenvolvimento para o grupo RUP

Número do Tipo	Nome do tipo	Quantidade de erros					
10	**Documentação**	**10.1**	**10.2**				
Total	0						
20	**Sintaxe**	**20.1**	**20.2**	**20.3**	**20.4**		
Total	2				2		
30	**Pacote**	**30.1**	**30.2**	**30.3**			
Total	0						
40	**Associação**	**40.1**	**40.2**	**40.3**	**40.4**		
Total	0						
50	**Interface**	**50.1**	**50.2**	**50.3**			
Total	4	3		1			
60	**Checagem**	**60.1**	**60.2**				
Total	0						
70	**Dados**	**70.1**	**70.2**				
Total	0						
80	**Funções**	**80.1**	**80.2**	**80.3**	**80.4**	**80.5**	**80.6**
Total	6	6					
90	**Sistema**	**90.1**	**90.2**	**90.3**			
Total	0						
100	**Ambiente**	**100.1**	**100.2**	**100.3**	**100.4**		
Total							
TOTAL	12						

As atividades do grupo XP foram cronometradas conforme a Tabela 3.

Tabela 3: Atividades e tempo de desenvolvimento do grupo XP

Data	**Atividades**	**Tempo (Minutos)**
05/03/03		405
	Criação das User Stories	
	Inicio da implementação das Users Stories	
	Inicio da criação dos Scripts de tests	
06/03/03	Alteração da interface	450
	Codificação	
	Testes de Unidade	
07/03/03	Alteração de Interface	405
	Alteração dos Scripts de Testes	
10/03/03		450
	Alteração de Interface	
	Codificação	
	Testes	
11/03/03		450
	Codificação	
	Testes Finais	
5 DIAS	**TEMPO TOTAL (MINUTOS)**	2160

As atividades do grupo RUP foram cronometradas conforme a Tabela 4.

Tabela 4: Atividades e tempo de desenvolvimento do grupo RUP

Data	Atividades	Tempo (Minutos)
05/03/03		405
	Documentação/Modelagem	
	Inicio da codificação e desenvolvimento da interface	
06/03/03	Documentação técnica	450
	Alteração da interface	
	Codificação	
	Testes de Unidade	
07/03/03	Alteração de Interface	360
	Codificação	
	Testes de Unidade	
10/03/03		450
	Alteração de Interface	
	Codificação	
	Testes de Unidade	
11/03/03		450
	Testes finais	
	Término e revisão da documentação/modelagem	
5 DIAS	**TEMPO TOTAL (MINUTOS)**	2115

Ao término da fase de implementação os softwares produzidos pelo grupo de XP e RUP, passaram para a equipe de testes que foram responsáveis pelos testes de aceitação.

3.2.5 Equipe de Testes

A equipe de testes foi composta por um analista de testes, analista de negócios e uma pessoa de apoio para coleta dos dados. Esta equipe teve a

responsabilidade de montar o plano e especificações de testes, executá-lo e coletar os resultados.

As *user stories* do XP e os *uses cases* do RUP foram utilizados para a elaboração do plano de testes e o mesmo plano foi usado para direcionar os testes do produto de software dos dois grupos. O relatório com a descrição do plano de testes do software está no Anexo VI.

Durante a execução do plano de testes os erros apresentados foram categorizados conforme HUMPHREY(1995) e os tipos de erros padrões apresentados no Quadro 4.

Quadro 4: Índice dos tipos de erros utilizados nos testes

10	**Associação**	10.1 – Declaração
		10.2 - Nomes duplicados
		10.3 – Escopo
		10.4 – Limites
20	**Interface**	20.1 - Chamadas de procedimentos e referências
		20.2 - I/O
		20.3 – Formatos
30	**Checagem**	30.1 - Mensagens de erros
		30.2 - Verificações inadequadas
40	**Funções**	40.1 – Lógica
		40.2 – Ponteiros
		40.3 – Loops
		40.4 – Recursividade
		40.5 – Computação
		40.6 – Defeitos de funções
50	**Sistema**	50.1 – Configuração
		50.2 – Tempo

A quantidade de erros encontrados no software produzido pelo grupo de XP, estão apresentados na Tabela 5.

Tabela 5: Quantidade de erros encontrados nos testes ao software do grupo XP

ID	Identificador	Caso de teste	Número do tipo de erro	Quantidade de erros
1	CISSPODER-RT-01-IT-01	CISSPODER-ETF-01-CT-04	20.3	2
2	CISSPODER-RT-01-IT-02	CISSPODER-ETF-05-CT-01	40.6	1
3	CISSPODER-RT-01-IT-03	CISSPODER-ETF-01-CT-02	40.1	1
4	CISSPODER-RT-01-IT-04	CISSPODER-ETF-01-CT-03	40.1	1
5	CISSPODER-RT-01-IT-05	CISSPODER-ETF-03-CT-05	10.4	1
6	CISSPODER-RT-01-IT-06	CISSPODER-ETF-03-CT-04	10.4	1
7	CISSPODER-RT-01-IT-07	CISSPODER-ETF-04-CT-01	40.1	1
8	CISSPODER-RT-01-IT-08	CISSPODER-ETF-04-CT-08	40.1	1
9	CISSPODER-RT-01-IT-09	CISSPODER-ETF-04-CT-09	30.2	1
10	CISSPODER-RT-01-IT-10	CISSPODER-ETF-04-CT-11	30.2	1
11	CISSPODER-RT-01-IT-11	CISSPODER-ETF-04-CT-12	20.1	1
12	CISSPODER-RT-01-IT-12	CISSPODER-ETF-05-CT-04	10.3	1
13	CISSPODER-RT-01-IT-13	CISSPODER-ETF-05-CT-03	10.3	1
14	CISSPODER-RT-01-IT-14	CISSPODER-ETF-06-CT-03	30.2	1
15	CISSPODER-RT-01-IT-15	CISSPODER-ETF-06-CT-04	30.2	1
16	CISSPODER-RT-01-IT-16	CISSPODER-ETF-07-CT-01	40.6	1
17	CISSPODER-RT-01-IT-17	CISSPODER-ETF-07-CT-02	40.6	1
18	CISSPODER-RT-01-IT-18	CISSPODER-ETF-07-CT-03	10.3	1
19	CISSPODER-RT-01-IT-19	CISSPODER-ETF-07-CT-04	10.3	1
20	CISSPODER-RT-01-IT-20	CISSPODER-ETF-07-CT-05	20.1	1
21	CISSPODER-RT-01-IT-21	CISSPODER-ETF-07-CT-06	20.1	1
			Total	22

A quantidade de erros encontrados no software produzido pelo grupo de RUP estão apresentados na Tabela 6.

Tabela 6: Quantidade de erros encontrados nos testes ao software do grupo RUP

ID	Tipo teste	Categoria	Número do tipo de erro	Quantidade de erros
1	CISSPODER-RT-01-IT-01	CISSPODER-ETF-01-CT-04	20.3	1
2	CISSPODER-RT-01-IT-02	CISSPODER-ETF-01-CT-05	20.3	1
3	CISSPODER-RT-01-IT-03	CISSPODER-ETF-06-CT-04	30.2	1
4	CISSPODER-RT-01-IT-04	CISSPODER-ETF-06-CT-01	20.3	1
5	CISSPODER-RT-01-IT-05	CISSPODER-ETF-01-CT-04	20.3	1
			Total	5

O tempo que foi cronometrado para equipe de testes executar os testes no software produzido pelo grupo de XP, está descrito na Tabela 7.

Tabela 7: Atividades da equipe de testes ao software do grupo XP

Data	Atividades	Tempo (Minutos)
18/03/03		
	Configuração do sistema para testes	15
	Teste Requisito Selecionar opção	20
	Teste Requisito Informar Vendedor	5
	Teste Requisito Informar Período	4
	Teste de Requisito Alterar lançamento de comissão	8
	Teste de Requisito Excluir lançamento de comissão	8
	Teste de Requisito Incluir lançamento de comissão	10
	Teste de Requisito Atualizar saldo de comissão	15
1 dia	**TEMPO TOTAL (MINUTOS)**	**85**

O tempo que foi cronometrado para equipe de testes executar os testes no software produzido pelo grupo de RUP, esta descrito na Tabela 8.

Tabela 8: Atividades da equipe de testes ao software do grupo RUP

Data	Atividades	Tempo (Minutos)
19/03/03		
	Configuração do sistema para testes	10
	Teste Requisito Selecionar opção	6
	Teste Requisito Informar Vendedor	2
	Teste Requisito Informar Período	2
	Teste de Requisito Alterar lançamento de comissão	7
	Teste de Requisito Excluir lançamento de comissão	3
	Teste de Requisito Incluir lançamento de comissão	4
	Teste de Requisito Atualizar saldo de comissão	8
1 dia	**TEMPO TOTAL (MINUTOS)**	**42**

Após a conclusão da primeira iteração ou ciclo de desenvolvimento pelas equipes de XP, RUP e testes, foram produzidos os relatórios de problemas encontrados e incidentes de testes relativo ao software produzido por cada um dos grupos. Estes relatórios podem ser verificados no Anexo IV. Os grupos de desenvolvimento com estes relatórios em mãos deram início ao segundo ciclo ou segunda iteração para que fossem corrigidos os erros detectados.

CAPÍTULO IV – RESULTADOS OBTIDOS

Após a conclusão das iterações de testes e acertos do software, os dados coletados foram tabulados e analisados como segue abaixo.

4.1 Análise do 1º Ciclo da Etapa de Desenvolvimento

Tabela 9: Comparativo de quantidade de erros e tempo de desenvolvimento para as duas metodologias

Metodologia	Quantidade Erros	Tempo em horas	Erros/hora
XP	21	36	0,61
RUP	12	35,25	0,34

Durante as atividades dos desenvolvedores de XP foram registrados 21 erros e nas atividades de RUP 12, perfazendo um índice de 0,61 e 0,34 erros por hora de desenvolvimento, respectivamente.

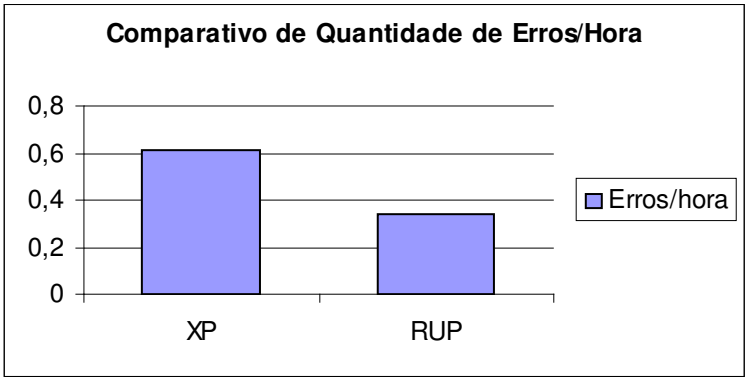


Figura 1: Comparativo da Quantidade de Erros por Hora de Desenvolvimento

A quantidade de erros encontrados durante o desenvolvimento e o tempo de implementação dos softwares a partir das duas metodologias aplicadas é mostrada na Figura 2. É perceptível que o tempo de desenvolvimento das equipes foi quase equivalente, porém a quantidade de erros encontrados na etapa para as equipes foi diferente, registrou-se uma diferença de 9 erros a mais para a equipe XP.

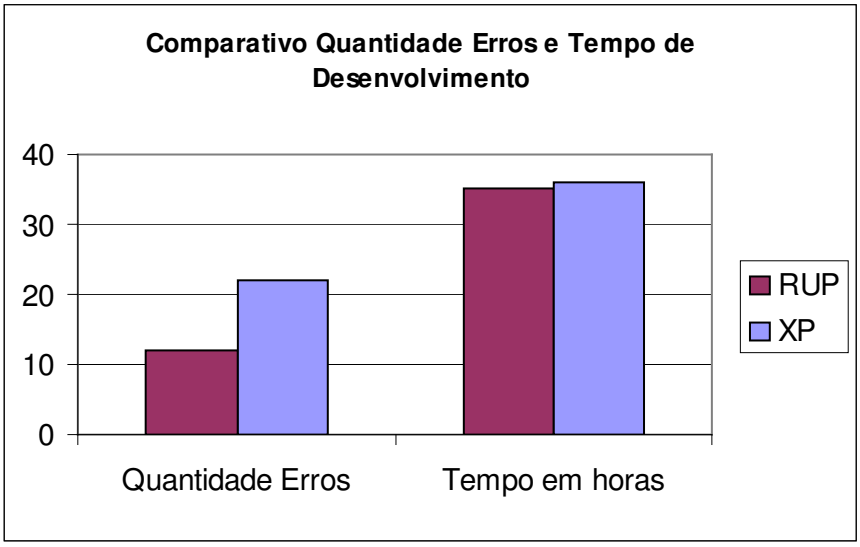


Figura 2: Comparativo Quantidade Erros e Tempo de Desenvolvimento

4.1.1 Análise dos Erros por Linhas de Código (LOCs)

Esta métrica, também conhecida como LOC conta todas as linhas de código executável, ignorando comentários no software desenvolvido.

A contagem do número de linhas dos códigos fonte dos softwares desenvolvidos foi feita a partir do Software Visual Source Safe 6.0. Foram contabilizadas 1695 linhas para o software do grupo XP e 1433 para o software do grupo RUP.

A partir destes números chega-se a uma comparação com a quantidade de erros encontrados na fase de testes, demonstrado na Tabela 10.

Tabela 10: Número de linhas de código, quantidade de erros e índice de erros por linha de código.

Metodologia	Linhas Código	Quantidade erros	Erros por linha de código	Erros por linha de código (%)
RUP	1433	5	0,003	0,349
XP	1695	22	0,01	1,30

Para cada metodologia o índice de erros por linha de código é compatível com a quantidade de erros encontrados; sendo 0,1 para XP e 0,003 para RUP.

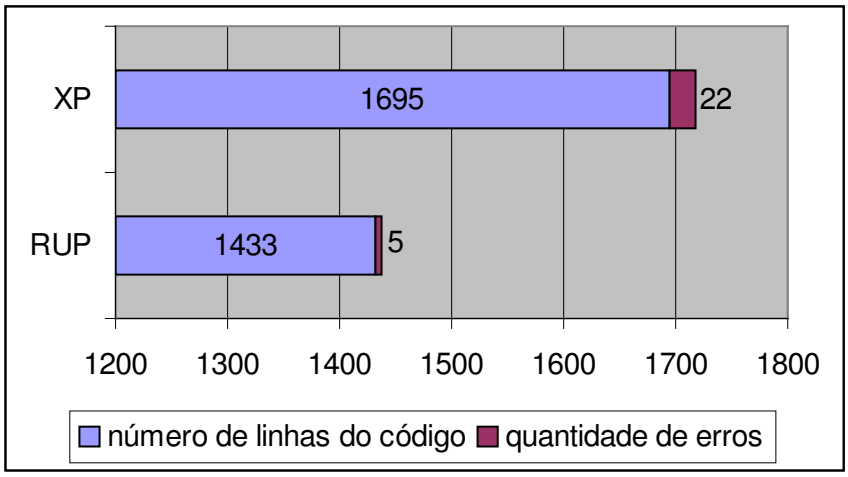


Figura 3: Quantidade de Linhas de Código e Erros– XP e RUP

4.2 Análise do 1º ciclo da Etapa de Testes

Na etapa de testes do primeiro ciclo foram aferidos a quantidades de erros e o tempo necessário para realização dos testes, apresentados na tabela 11.

Tabela 11: Comparativo de quantidade de erros e tempo de testes para as duas metodologias

Metodologia	Quantidade erros	Tempo Teste Metod.(Horas)	Erros/hora
RUP	5	0,7	7,14
XP	22	1,4167	15,53

Durante a fase de testes foram registrados 22 erros para o software XP e 5 erros para o software RUP, gerando o índice de 7,14 e 15,53 erros por hora de testes, respectivamente e mostrados na Figura 4.

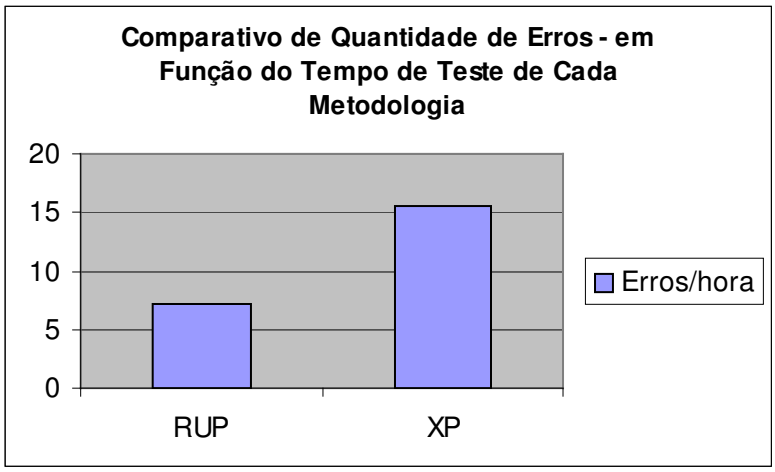


Figura 4: Comparativo da Quantidade de Erros por Hora de Teste

A quantidade de erros encontrados durante os testes e o tempo cronometrado para as duas metodologias aplicadas é mostrada na Figura 5.

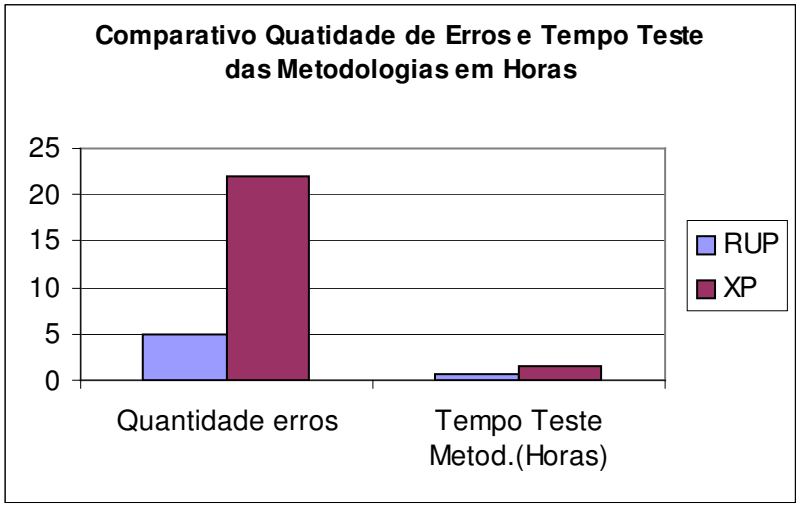


Figura 5: Comparativo Quantidade Erros e Tempo de Testes

Com base no índice de horas, verificou-se que com o tempo de testes de RUP(0,7 horas) foram encontrados 5 erros enquanto que com o dobro deste tempo de testes de XP(1,4167 horas) foram encontrados 22 erros. Portanto, com o esforço de tempo de testes em XP registrou-se um índice de erros por hora superior ao índice de erros por hora de RUP, pois se com 0,7 horas encontrou-se 5 erros, com 1,4167 horas equivalentemente encontrar-se-iam de 10 a 12 erros, mas foram registrados 22 erros.

4.2.1 Tipos de Erros Registrados no 1º Ciclo de Desenvolvimento

Dentro do ciclo de desenvolvimento dos softwares com as duas metodologias propostas, foram registrados diferentes tipos de erros para ambas as equipes.

Estes erros foram categorizados como mostrou o Quadro 3, assim a partir deste quadro foram elaborados gráficos para demostrar a contabilização de erros, para cada categoria e suas sub-categorias, das equipes XP e RUP.

Tabela 12: Quantidade de erros por categoria – XP e RUP

Nome do tipo	Quantidade XP	Quantidade RUP
Documentação	0	0
Sintaxe	11	2
Pacote	0	0
Associação	2	0
Interface	0	4
Checagem	0	0
Dados	0	0
Funções	8	6
Sistema	0	0
Ambiente	0	0

A Tabela 12 mostra que ocorreram 12 erros na categoria Sintaxe, 2 erros na categoria Associação e 8 erros na categoria Funções para a equipe XP, não sendo registrados erros nas demais categorias. A Tabela 12 também mostra a ocorrência de 2 erros na categoria Sintaxe, 4 erros na categoria Interface e 6 erros na categoria Funções para a equipe RUP, não sendo registrados erros nas demais categorias.

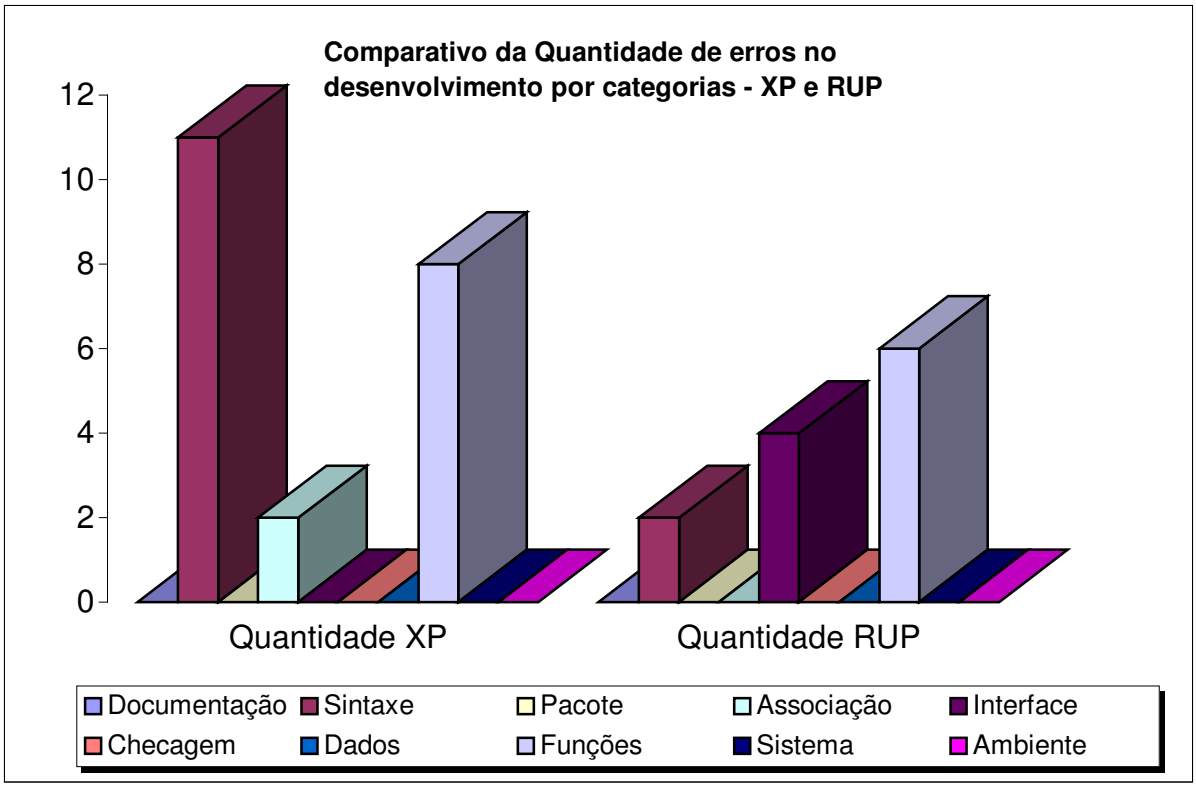


Figura 6: Quantidade de erros no desenvolvimento por categorias – XP e RUP 1º ciclo

Conforme mostra o gráfico acima, ocorreram erros de Associação para a equipe XP mas que não ocorreram para a equipe RUP, assim como, ocorreram erros de Interface para a equipe RUP e não ocorreram para a equipe XP. Os erros de Sintaxe e Funções são comuns às duas equipes, porém em quantidades diferentes, conforme mostra a Tabela 12.

A partir das categorias, foram especificados os tipos de erros registrados nas sub-categorias, assim a Tabela 13 mostra os erros ocorridos nas equipes XP e RUP, respectivamente relativo às sub-categorias.

Tabela 13: Quantidade de erros no desenvolvimento por sub-categorias – XP e RUP

Sub-categorias	Quantidade XP	Quantidade RUP
10.1 – Comentários	0	0
10.2 – Mensagens	0	0
20.1 – Escrita	4	0
20.2 – Pontuação	0	0
20.3 – Tipos	1	0
20.4 – Formato das instruções	6	2
30.1 - Gerenciamento mudança	0	0
30.2 - Bibliotecas de programas	0	0
30.3 – Controle de versão	0	0
40.1 – Declaração	2	0
40.2 - Nomes duplicados	0	0
40.3 – Escopo	0	0
40.4 – Limites	0	0
50.1 - Chamadas de procedimentos e referências	0	3
50.2 - I/O	0	0
50.3 – Formatos	0	1
60.1 - Mensagens de erros	0	0
60.2 - Verificações inadequadas	0	0
70.1 – Estruturas	0	0
70.2 – Conteúdos	0	0
80.1 – Lógica	8	6
80.2 – Ponteiros	0	0
80.3 – Loops	0	0
80.4 – Recursividade	0	0
80.5 – Computação	0	0
80.6 – Defeitos de funções	0	0
90.1 – Configuração	0	0
90.2 – Tempo	0	0
90.3 – Memória	0	0
100.1 – Projeto	0	0
100.2 – Compilação	0	0
100.3 – Teste	0	0
100.4 – Outros problemas de suporte de sistemas	0	0

Em relação à Tabela 13 que mostra os erros ocorridos durante a fase de desenvolvimento para a equipe XP e a equipe RUP por sub-categorias, verifica-se que ocorreram mais erros para a equipe XP na categoria 20 (Sintaxe) – sendo 11 do total de 21 erros -, já para a equipe RUP as quantidades foram registradas para sub-categorias de categorias diferentes – sendo 2 erros na categoria 20(Sintaxe), 4 erros na categoria 50(Interface) e 6 erros na categoria 80(Funções).

4.3 Análise do 2º Ciclo da Etapa de Desenvolvimento

Durante as atividades dos desenvolvedores de XP no 2º ciclo foram registrados 9 erros e nas atividades de RUP 4, perfazendo um índice de 1,93 e 1,03 erros por hora de desenvolvimento, respectivamente. As atividades de desenvolvimento do 2º ciclo foram para alterar o software, corrigindo os incidentes encontrados durante a fase de testes do 1º ciclo.

Tabela 14: Comparativo da quantidade de erros e tempo de desenvolvimento para as duas metodologias

Metodologia	Quantidade Erros	Tempo em horas	Erros/hora
XP	9	3,9	1,93
RUP	4	4,67	1,03

Em comparação com o tempo de desenvolvimento do 1º ciclo foi criado o gráfico da Figura 7:

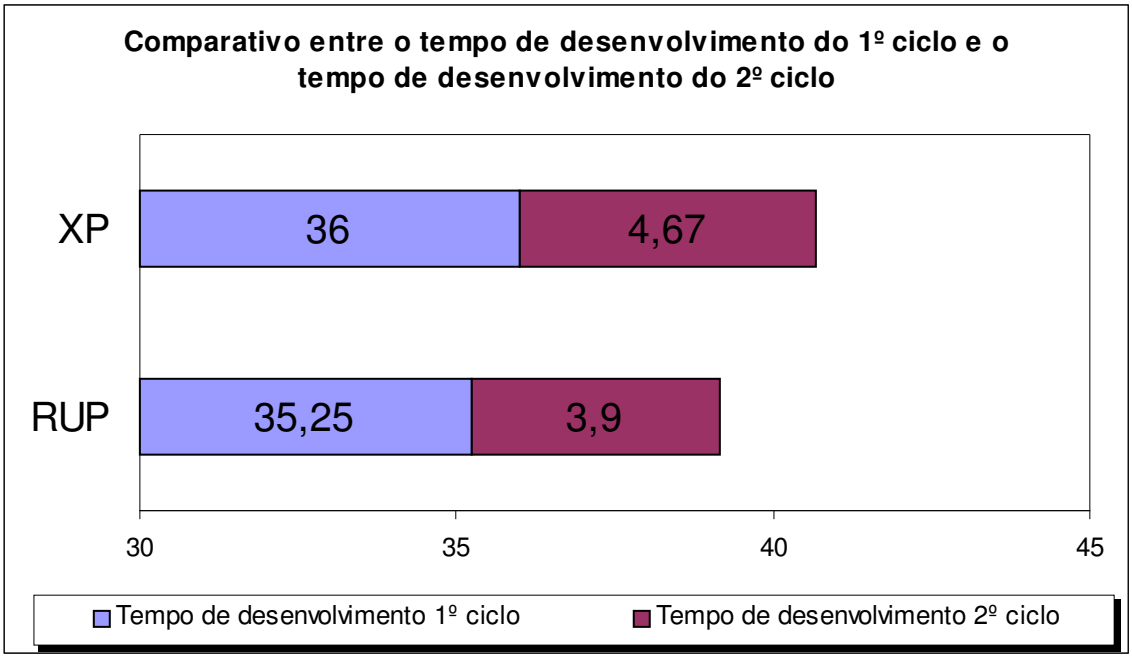


Figura 7: Comparativo entre os tempos de desenvolvimento do 1º e 2º ciclo.

4.4 Análise do 2º ciclo da Etapa de Testes

Nas atividades de testes do 2º ciclo foram executadas as especificações de testes do plano de teste utilizado anteriormente, assim os testes de regressão foram executados concomitantemente para o grupo XP e para o grupo RUP.

Os incidentes ocorridos nos testes do 1º ciclo, para o grupo XP e para o grupo RUP, foram determinados como corrigidos, pois não ocorreram nos testes de regressão executados com o plano de testes.

CAPÍTULO V – CONCLUSÃO

5.1 Conclusão

Após a análise isolada dos dados obtidos pela coleta, partiu-se para uma avaliação conjunta dos resultados, o que proporcionou as conclusões a seguir elencadas.

No tocante ao tempo total necessário para o desenvolvimento, testes e entrega do produto de software, utilizando-se cada uma das metodologias observou-se que a metodologia RUP e a XP apresentaram resultados praticamente equivalentes, sendo 39 e 40 horas respectivamente. A justificativa para estes resultados está no fato que mesmo dedicando um tempo considerável na metodologia RUP para a modelagem e documentação, atividades que não se aplicam no XP, a atividade de criação de scripts de testes no XP fizeram com que o resultado final fosse praticamente idêntico.

Em relação à ocorrência de erros, notou-se que nas etapas de projeto e implementação do 1º ciclo, o grupo de XP apresentou 42% a mais que o grupo de RUP. Mesmo tendo a vantagem que o custo de um erro detectado nestas fases ser inferior ao que um detectado pela equipe de testes ou até mesmo no cliente, o que chama atenção é o fato da quantidade de erros produzidos pela equipe XP. Estes dados poderiam ser vistos por uma outra ótica, que devido aos scripts de testes

feitos pelos desenvolvedores XP antes da implementação definitiva a quantidade erros detectados nesta etapa seria maior que a do RUP, mas esta visão não se confirmou devido estas proporções de erros terem se mantido na etapa dos testes de aceitação realizados pela equipe de testes.

Comparando-se a aplicação das duas metodologias, identificou-se que a XP obteve um maior custo/benefício, tendo como base que os esforços de tempo necessário para o desenvolvimento e testes foram praticamente os mesmos e produziram um resultado final de menor qualidade, devido aos erros serem em maior quantidade que no RUP. Uma prova disso é a quantidade de erros por linhas de código. Além da equipe RUP conseguir um código mais enxuto, com menor número de linhas, o índice de erros por linha obtido por eles foi aproximadamente três vezes menor se comparado ao da equipe de XP.

Os dados resultantes da pesquisa levam a concluir que a utilização da metodologia RUP é mais adequada, levando-se em consideração que o objetivo era saber qual das duas metodologias proporciona o desenvolvimento de um software de melhor qualidade, com o menor custo possível. Porém, existem algumas considerações que precisam ser feitas a esse respeito. Não é prudente e nem possível efetuar uma conclusão genérica, e sim, afirmar que neste caso, sob estas condições de parâmetros variáveis e escopo apresentado do projeto, obteve-se melhor resultado com a aplicação das técnicas e práticas sugeridas pelo RUP.

Outra observação importante é a de que a pesquisa efetuou apenas um corte, coletando dados estáticos medindo um momento em que as duas metodologias estavam sendo utilizadas pelas equipes sem uma grande experiência anterior em sua aplicação, pois esse era o objetivo e escopo deste trabalho. Existe uma hipótese, no entanto, que através da prática do uso das metodologias, os processos ficariam mais solidificados entre as equipes, podendo proporcionar os mesmos ou diferentes resultados numa segunda avaliação. Práticas orientadas por uma e por outra metodologia influenciariam no longo prazo, como por exemplo a disseminação do conhecimento proporcionado pela programação em pares.

5.2 Recomendações

Como sugestão para trabalhos futuros, recomenda-se repetir a experiência em projetos de maiores proporções, tanto ao nível de pessoas envolvidas quanto ao nível da quantidade de requisitos do software.

Outra recomendação seria aplicar o mesmo experimento mas mesclando algumas práticas de RUP E XP, comparando com os resultados da aplicação original.

BIBLIOGRAFIA

BECK, Kent., **Extreme Programming Explained: Embrace Change**, 2000.

BOOCH, Grady e RUMBAUGH, James e JACOBSON, Ivar. **UML Guia do Usuário**. Ed. Campus: Rio de Janeiro, 2000.

FEIGENBAUM, E.A. e P.MCCORDUCK. **The Fifth Generation.** Addison-Wesley, 1983.

FILHO, Wilson. P. P. **Engenharia de Software: Fundamentos**, **Métodos e Padrões**. Rio de Janeiro: Ed. LTC 2001

FINKELSTEIN, Anthony e KRAMER, Jeff. **Software Enginnering: A Roadmap.** 2002. Departamento de Ciência da Computação da Universidade de Londres

FOWLER, Martin. **The New Methodology,** 2000 . Disponível em http://www.martinfowler.com/articles/newmethodology.html. Acesso em 05 março 2002.

GHEZZI, Carlo e JAZAYERI, Medi e MANDRIOLLI, Dino. **Fundamentals of Software Engineering.** Ed. Prentice – Hall do Brasil, RJ: 1991.

Guide to the Software Engineering Body of Knowledge: trial version (version 1.00) executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis, Leonard L. Tripp. p. cm. 2001

HAYES, Steve. **An Introduction to Extreme Programming**, 2001. Disponível em http://www.khatovartech.com. Acesso em 15 abril 2002.

HIGHSMITH, Jim. **History**: **The Agile Manifesto**. 2001, Disponível em http://www.agilealliance.org. Acesso em 24 de fevereiro de 2002.

HUMPHREY, Watts S. **A Discipline for Software Engineering.** Reading MA: Addison – Wesley Longman, Inc, 1995.

JEFFRIES, Ron. **What is Extreme Programming.** Disponível em http://www.extremeprogramming.com. Acesso em 20 fev. 2002.

LARMAN, Craig. **Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos.** Porto Alegre: Bookman, 2000.

LONGMAN , Addison Wesley, **XP Immersion***,* 1998. Disponível em http://www.objectmentor.com. Acesso em 05 de maio de 2002.

MAFFEO, Bruno **Engenharia de Software e Especificação de Sistemas.** Rio de Janeiro: Ed. Campos, 1992

MOLINARI, **Leonardo, Testes de Software: Produzindo Sistemas Melhores e Mais Confiáveis.** São Paulo: Ed. Érica 2003.

STAIR, R. M, **Princípios de Sistemas de Informação.** Rio de Janeiro: Ed. LTC 1998.

OSBORNE, A., Running Wild. **The Next Industrial Revolution**. Osborne/McGraw-Hill, 1979.

POLLICE, Gary **Using the Rational Unified Process for Small Projects: Expanding Upon Extreme Programming.** RATIONAL SOFTWARE CORPORATION, 2001

PRESSMAN, R. S. . **Engenharia de Software,** Makron Books, 1995.

QUATRANI, Terry. **Modelagem Visual com Ration Rose 2000 e UML.** Rio de Janeiro: Ed. Ciência Moderna Ltda, 2001.

RATIONAL SOFTWARE CORPORATION. **Rational Unified Process – Best Practices for Software Development Teams.** Disponível em http://www.rational.com. Acesso em 08 abril 2002.

RIEHLE, Dirk. **A Comparison of the Value Systems of Adaptive Software Development and Extreme Programming: How Methodologies May Learn from Each Other**. 2000. Disponível em www.skyva.com, http://www.skyva.de. Acesso em 10 março 2002.

SILVA, Alberto M. Rodrigues e VIDEIRA, Carlos A Escaleira. **UML, , Metodologias e Ferramentas CASE.** 1.ed. Portugal: Centro Atlântico, 2001.

SOMMERVILLE, Iam. **Software Engineering.** 5th Ed, Addison Wesley, 1995

WAKE, William C.. **Extreme Programming Explored ,** 2000.

ANEXOS

Anexo I - Modelagem do negócio

As empresas de uma maneira geral trabalham com a política de comissionamento de suas equipes de vendas, desta forma se faz necessário um controle das operações de débito, crédito e também o saldo destas comissões.

Este módulo a ser desenvolvido terá a finalidade de permitir ao usuário lançar valores de comissão para os vendedores cadastrados no sistema, deverá ser considerado que já existe no sistema o recurso de cadastramento dos vendedores .

Deverá conter também as funcionalidades de inclusão, pesquisa, alteração e exclusão de lançamentos de comissões, bem como a atualização do saldo de comissão dos vendedores na tabela comissao_saldo. A gravação dos lançamentos será feita na tabela de comissao_movimento.

O módulo desenvolvido deverá ter as seguintes características de interface:

- *Tela de Pesquisa:* pesquisa e exclusão dos lançamentos;
- *Tela de Lançamento:* inclusão e alteração dos lançamentos;

Inclusão de lançamento contendo:

Campos obrigatórios para o usuário: código do vendedor, data de movimento, natureza (C/D), valor base da comissão, porcentagem da comissão (campo calculável quando informado o valor da comissão) e valor da comissão (campo calculável quando informado a porcentagem de comissão).

Campos opcionais para o usuário: código do cliente/fornecedor da nota referenciada, nº da nota, série da nota e observação do lançamento, origem do lançamento (Avulso), tipo do movimento (A vista/ A prazo), código da empresa.

Campos obrigatórios para visualização: saldo de comissão do dia e saldo atual de comissão (todos os dias de lançamentos).

O código da empresa deverá vir preenchido com a empresa corrente do sistema, e a data de movimento preenchida com a data atual do sistema operacional da máquina.

Para efetuar a gravação de um lançamento, pelo menos os campos obrigatórios deverão estar preenchidos antes de acionar o botão salvar ou teclas Ctrl + S. Se os campos obrigatórios não estiverem todos preenchidos o sistema emitirá mensagem solicitando preenchimento do(s) campo(s) que o usuário não preencheu.

Exclusão de lançamento: a exclusão de lançamento será solicitada pelo usuário na tela de pesquisa, mediante a mesma, o sistema pedirá confirmação da exclusão. Para o usuário efetuar uma exclusão, o lançamento desejado deverá estar selecionado antes de acionar o botão excluir ou teclas Ctrl + X.

Alteração de lançamento: a alteração de lançamento será solicitada pelo usuário na tela de pesquisa pelo duplo-clique no lançamento ou acionamento do botão “Alterar” estando o lançamento selecionado, assim, os dados contidos neste lançamento preencherão os campos referentes na tela de lançamento. O usuário poderá alterar qualquer campo já preenchido (campos obrigatórios), como também preencher campos que não estavam (campos opcionais). Para efetuar a gravação de um lançamento alterado, todos os campos obrigatórios deverão estar preenchidos antes de acionar o botão salvar ou teclas Ctrl + S. Se os campos obrigatórios não estiverem todos preenchidos o sistema emitirá mensagem solicitando preenchimento do(s) campo(s) que o usuário não preencheu.

Pesquisa de lançamento: a pesquisa disponibilizará campos para filtragem dos lançamentos: código da empresa, data inicial e data final dos lançamentos, código do vendedor, nº da nota, série da nota, valor inicial e valor final das comissões dos lançamentos, natureza (Débito/Crédito). Estes campos foram disponibilizados para os desenvolvedores optarem quais seriam mais adequados para a filtragem. O código da empresa deverá vir preenchido com a empresa corrente do sistema, e a data inicial e data final preenchidas com a data atual do sistema operacional da máquina. Após preenchimento dos campos para filtragem, o usuário deverá acionar um botão para a pesquisa ser efetuada, e os lançamentos serão mostrados na tela.

Anexo II - Artefatos de saída originados das fases do grupo RUP

Requisitos
Objetivo Geral:
Desenvolver um módulo de Comissões Avulsas, com a possibilidade do cliente, fazer o lançamento avulsamente das comissões, incluindo débitos e créditos e também verificando o saldo da comissão de cada vendedor.

Objetivos específicos e funções do sistema
1. Ter a possibilidade de incluir comissões por vendedor, ou seja ao entrar na tela poderá informar o vendedor que será lançada a comissão.
2. Também poderá informar a data de lançamento de comissão. Para lançamento com datas retroativas é necessário criar um processo para que o usuário possa ou não ter acesso a fazer esse tipo de lançamento.
3. O usuário poderá dizer qual a natureza da comissão: se é débito ou se é crédito.
4. Esses campos deverão ser obrigatórios e se não preenchidos deverá emitir mensagem ao usuário para preenchê-los.
5. O débito irá reduzir a comissão e será representado pela letra “D”
6. O Crédito aumentará a comissão e será representado pela letra “C”
7. O usuário poderá informar o valor da base que será calculada a comissão. Ex: 100,00 * 10%(percentual de comissão) = 10,00(valor de comissão)
8. A comissão poderá ser informada de duas formas em percentual e em valor.
9. Se informado em percentual o valor da comissão deve ser preenchido automaticamente e se informado o valor da base o percentual deverá ser preenchido automaticamente.
10. O valor da comissão deve ser um campo obrigatório, quando o usuário salvar, e este campo não estiver preenchido, deverá emitir mensagem que o campo não foi preenchido e voltar no campo para preenchê-lo.
11. Também deverá existir os campos: Cliente, nº da nota fiscal ou cupom, descrição do lançamento, esses campos são opcionais.
12. No caso em que o cliente informa os campos acima é quando ele possui uma nota fiscal ou um cupom que não registrou comissão ou registrou a comissão incorreta e ele deseja fazer o acerto.
13. É interessante um campo que mostre quanto o vendedor possui de saldo de comissões até o momento, e quando o usuário salvar suas alterações deverá atualizar esse saldo automaticamente.
14. O usuário também terá a possibilidade de excluir e alterar lançamentos, isso poderá ser feito na aba de pesquisa, onde ele visualizará o lançamento, com duplo clique ou através de um botão alterar ele poderá alterar o lançamento feito anteriormente, quando pressionado em alterar, ou dado duplo clique o lançamento deve ser carregado na aba de lançamento.
15. A alteração do lançamento deverá atualizar o saldo automaticamente na gravação.
16. Quando o usuário excluir o lançamento o saldo deverá ser atualizado automaticamente.
17. O usuário poderá imprimir a movimentação mostrada na tela de pesquisa.

18. Na tela de pesquisa o usuário terá a opção de informar a empresa, o vendedor o período, e o valor do lançamento. A empresa já virá preenchida com a empresa atual, o usuário poderá consultar mais de uma empresa ao mesmo tempo e também deixar os outros campo em branco para consultar toda a movimentação.
19. Na tabela de comissao_movimento, o tipoorigempercentual deverá ser preenchido com A – avulso.
20. A pesquisa mostrará apenas os lançamentos avulsos, mas o Saldo do dia e atual será de toda movimentação da tabela de comissao_movimento.

Atributos

Tempo de resposta:

- Para entrar na tela: 2s
- Para pesquisa: mínimo 2s, máximo: 5s

Tolerância a falha:

- Confirmação na exclusão de um lançamento.
- Emitir mensagem que o usuário não têm permissão para fazer lançamento no caso da data retroativa. (conforme processo)
- Os relatórios do sistema deverão ser ajustados para que todos peguem informações das tabelas comissao_movimento e comissão_saldo, porque senão vai ocorrer diferenças com os relatórios que pegam informações da tabela de estoque_analitico.

Plataforma:

- Todos os sistemas operacionais Windows
- Impressão da consulta: Hardware: Impressoras Office/Desktop - Laser, Deskjet, BubbleJet;

Protótipo:

1. A tela deve ser composta por duas abas uma para lançamento de comissões e outra para pesquisa de comissões; (elas serão datawindows)
2. A tela de lançamento de comissões, ficará da seguinte forma:
   - O campo vendedor será um campo text com o ícone de pesquisa vendedores do lado, ao clicar no ícone deverá abrir a tela de pesquisa vendedores. (campo obrigatório), se não preenchido não habilita os outros campos para preenchimento.
   - O campo Natureza, será um dropdown, com a opção débito e crédito. (campo obrigatório)
   - O campo percentual de comissão será um spin
   - O campo cliente terá o ícone de localizar do lado, que ao pressioná-lo abrirá tela de pesquisa clientes.
   - Os outros campos serão text, sendo que os de valor terão máscara.
3. Na aba de Pesquisa o usuário terá vários filtros:

- A empresa e o vendedor, terão o ícone de localizar que ativará tela de pesquisa.
- O período inicial e final, terá o ícone de calendário do lado.
- O filtro valor, será com máscara.
- Terá os botões ok e cancelar

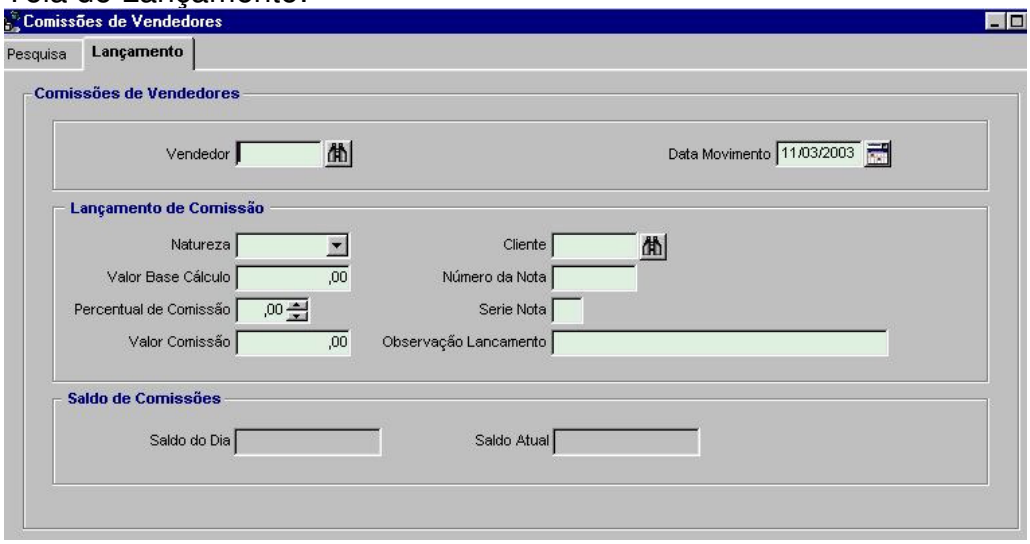
4. Na tela de pesquisa mostrará as seguintes colunas:

- Empresa, Data de movimento, Vendedor, valor da base, Débito, Crédito, Saldo, usuário, planilha, observação.

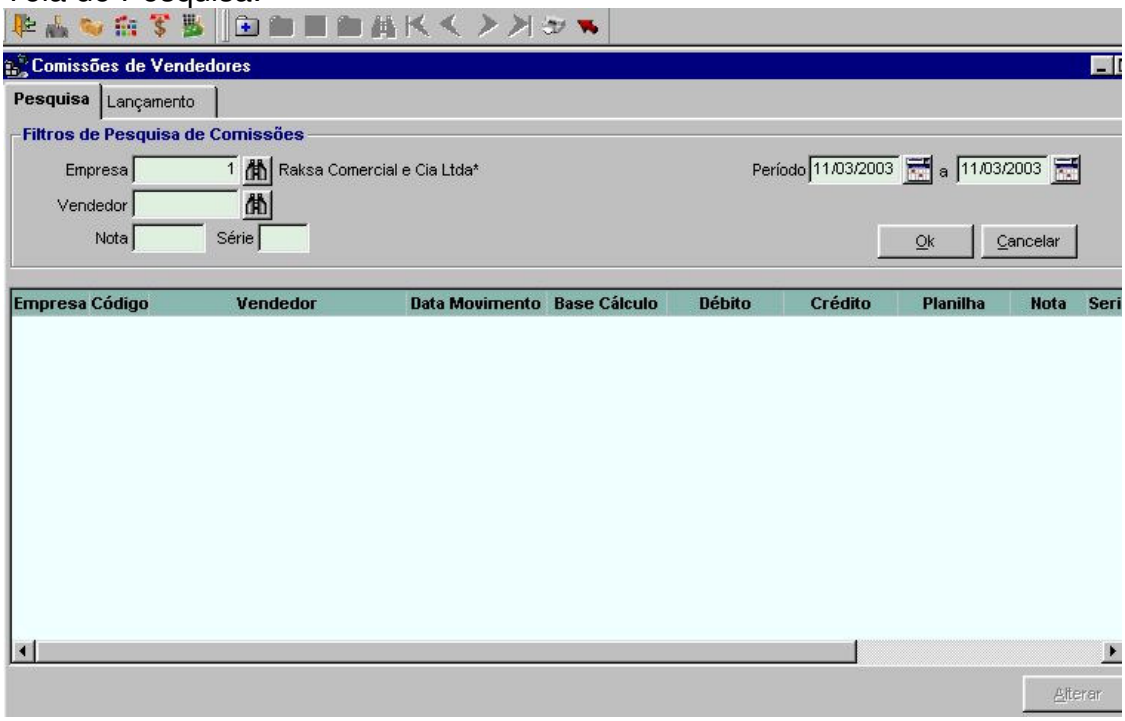
- Quando o usuário clicar nas colunas deverá ordenar conforme a coluna clicada.

5. Para alteração do lançamento o usuário terá o opção de dar duplo clique sobre o lançamento e também terá na tela abaixo um botão alterar, quando o usuário não estiver sobre nenhum lançamento este botão deverá ficar desabilitado.
6. Para exclusão do lançamento o usuário terá a opção do toolbar, ou pressionar ctrl+X sobre o lançamento.
7. As opções de incluir, excluir, cancelar do toolbar deverão ficar habilitadas ou desabilitadas conforme as funções.

Tela de Lançamento:



Tela de Pesquisa:



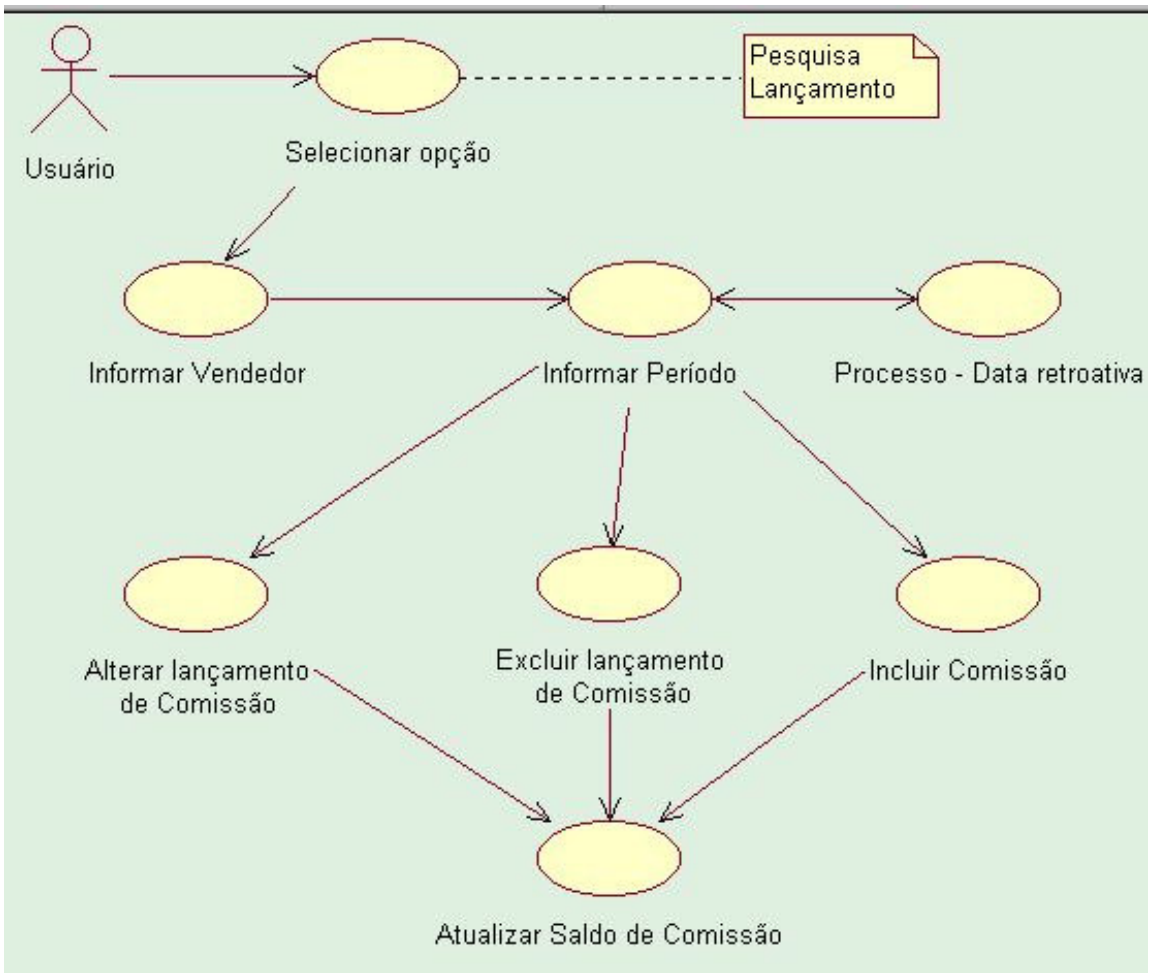
Definição de Casos de Uso

Atores: Usuário

Casos de Uso:

- Selecionar opção(Pesquisa, lançamento),
- informar vendedor,
- informar período de lançamento,
- Processo – Data retroativa
- Inclusão do lançamento de comissão,
- alteração do lançamento,
- exclusão do lançamento,
- atualização de saldo.

Diagrama de Caso de Uso



Base de Dados

Comissão_movimento	+ IDEmpresa + IDPlanilha + IDVendedor + IDClifor + Dtmovimento + NumNota + ObsLancamento + SerieNota + TipoMovimento + TipoNatureza + TipoOrigemPercentual + ValBaseCalculo + ValComissao
Nome: IDEmpresa Tipo: **ID - Chave Primária** Obs.: Identificador do codigo da empresa	Nome: IDPlanilha Tipo: **ID – Chave Primária** OBS: Identificador de planilha
Nome: IDVendedor Tipo: **ID – Chave Primária** **Obs: Codigo de identificação do vendedor no cadastro de vendedores.**	Nome: IDClifor Tipo: ID Obs: Codigo de identificacao do cadastro
Nome: Dtmovimento Tipo: Data Obs: Data em que ocorreu o lancamento	Nome: NumNota Tipo: Número Obs: Numero da nota ou cupom fiscal que originou o lancamento.
Nome: ObsLancamento Tipo: String Obs: Observacao que identifica o tipo dp o lancamento.	Nome: SerieNota Tipo: String Obs: Série do documento gerador
Nome: TipoMovimento Tipo: Tipo (texto) Obs: Identifica o movimento: A- a vista P- a prazo C- cancelado	Nome: TipoNatureza Tipo: Tipo (texto) Obs: Tipo da natureza do lancamento "C" – credito "D" – debito
Nome: TipoOrigemPercentual Tipo: Tipo (texto) Obs: Identifica a origem do percentual de comissao: "T" - tabela especial "F" - forma pagamento "P" – produto "V" – Vendedor “A” – Avulso	Nome: ValbaseCalculo Tipo: Valor Obs: Valor que serve como base para se calcular o VlrComissao
Nome: Valcomissao Tipo: Valor Obs: Valor da comissao ou estorno no lancamento	

Comissao_Saldo	+Idempresa +IdVendedor +Dtmovimento +ValTotCredito +ValTotDebito +ValSaldoAtual
Nome: IDEmpresa Tipo: ID – **Chave Primária** Obs: Identificador da empresa	Nome: IDVendedor Tipo: ID – **Chave Primária** Obs: Identifica o vendedor – cadastro de vendedores.
Nome: Dtmovimento Tipo: Data – **Chave Primária** Obs: Data de lançamento da comissão	Nome: ValTotCredito Tipo: Valor Obs: Valor de Creditos da comissão
Nome: ValTotDebito Tipo: Valor Obs: Valor de Débitos da Comissão	Nome: ValSaldo Atual Tipo: Valor Obs: Valor do saldo de comissão: valtotCredito – valtotdébito.

OBS: Os campos serão gravados na tabela acima, mas o saldo deverá ser atualizado na tabela de saldos. O saldo será atualizado automaticamente através de uma trigger no banco.
comissao_saldo / comissao_movimento
Relacionamento com outras tabelas:
Empresa
Cliente_fornecedor
Usuarios
Direitos_processos

Glossário

Comissão: É uma remuneração(descrita em valor ou percentual) que a empresa paga ao vendedor, por prestar um serviço a mesma.
Débito: natureza do lançamento e reduzirá o saldo de comissão
Crédito: natureza do lançamento e aumentará o saldo de comissão.
Base de Cálculo: É a base para se calcular o valor da comissão.

Documentação Técnica

- Criar uma aplicação nova, incluíndo a mesma no library através das propriedades.
- Criar uma window inherit da w_acestralcadastro
- Colocar uma tab, inicialmente com duas tabpages: Pesquisa e Lançamento
- Na tabpage pesquisa existirá duas datawindows: dw_filtros: datawindow que contém os filtros para pesquisa e a dw_pesquisa: datawindows que contém as colunas que serão mostradas na tela, após informar filtros. Essas datawindows são herdadas do ue_datawindow que se encontra no poder.pbl
- A tabpage de lançamento possui duas datawindow, as quais também herdarão da datawindow ue_datawindow
- Na tabpage de lançamento, o campo percentual de comissão não será gravado, mas será um **Cast decimal** na syntaxe da datawindow.
- Os campos idvendedor e idclifor, também serão um Cast com o nome de idvendedore e idclifore, mas no evento ue_validacao serão jogados no banco para respectivamente idvendedor e idclifor o conteúdo desses campos.

- O idempresa será gravado no banco a empresa atual.
- No campo Tipoorigempercentual será gravado no banco sempre “A” para identificar que o lançamento é Avulso.
- Criado um processo no objeto nv_atualiza, função: of_atualiza_processos, nº 111(poder.pbl).

TESTES

Teste de requisitos:

- Testado os requisitos descritos acima.

Teste de interface:

- Padrões do Sistema: ícones, botões, fontes, groupbox, campos obrigatórios em cor azul, datawindow, acentos, campos bloqueados, tabs, etc.

Teste Funcional:

- Implementado validações no ue_validação, para que na gravação, seja verificado se os campos obrigatórios estão preenchidos corretamente.
- O cálculo de comissão é realizado ao contrário também (ex: informado valor da comissão, e base de cálculo, para calcular o percentual). Implementado para que se o usuário deixar um dos campos em branco (base de cálculo por exemplo), seja emitido mensagem e evitando divide by zero.

Teste de Configuração:

- Testado o funcionamento do processo para bloquear por usuário, quando este informar data retroativa, tanto na inclusão do lançamento como na alteração.

Teste de Integridade:

- Testar através do dbisqlc, para verificar se o que foi informado foi gravado corretamente no banco. Verificado inclusão, alteração e exclusão, se atualizará os dados no banco.

Documentação (Help)

Caminho: **Menu de Controladoria/ opção Comissões de Vendedores**

Quando é realizado uma venda tanto na frente de caixa como no tele-vendas, se o vendedor possui comissão, e o COI de venda está marcado para gerar comissão, é guardado essa comissão gerada na tabela de Comissao_Movimento. Lá aparecem informações importantes, como o percentual de comissão e o valor de comissão de cada venda e de cada vendedor.

A opção de comissões de vendedores, serve para o usuário lançar novas comissões e ajustar esses mesmos lançamentos se assim desejar, tendo um controle do que entrou e do que saiu para cada vendedor, sendo uma opção de inclusão, alteração e exclusão de lançamentos avulsos de comissão de cada vendedor.

Aba de Lançamento

Essa aba serve para o usuário incluir e alterar lançamentos de comissões selecionados pela aba de pesquisa, podendo também visualizar o saldo diário e o saldo atual das **comissões.**

Campos dessa aba:

- **Vendedor:** Serve para o usuário informar o vendedor que será lançado a comissão, ou também para o usuário verificar o saldo de comissão desse vendedor.
- **Data de Movimento:** É a data de movimento que será lançada a comissão. No caso da data existe um processo, no seguinte caminho: Menu Cadastro/opção Complementares/opção de Direitos e processos de usuários/ processos/ pasta de Controladoria/ processo: Permite o usuário informar data retroativa para o lançamento de comissão. Se estiver marcado sim, o usuário poderá incluir ou alterar um lançamento de comissão informando uma data retroativa, se estiver como não, o usuário não poderá realizar lançamentos com data retroativa e também não poderá alterar e nem excluir os lançamentos já existentes.
- **Natureza:** Débito: Reduzirá o saldo de comissão/ Crédito: Aumentará o saldo de comissão.
- **Base de Cálculo:** É a base utilizada para o cálculo de comissão. Exemplo:
  Base de cálculo R$ 200,00
  Percentual de comissão: 5%
  Valor da comissão: 200,00 * 5% = 10,00
- **Percentual de Comissão:** É o percentual utilizado para calcular o valor da comissão, o usuário poderá deixar este campo em branco e informar somente o valor da base e o valor da comissão que o sistema automaticamente, mostrará o percentual de comissão. Exemplo:
  Base de cálculo R$ 200,00
  Valor da comissão: 200,00 * 5% = 10,00
  Percentual de comissão: (10,00 / 200,00)* 100 = 5%
- **Valor da Comissão:** Valor que será pago ao vendedor, é a base de cálculo vezes o percentual de comissão do vendedor. Será acrescido ou deduzido conforme a natureza(crédito/débito) do lançamento.
- **Cliente/ Número da nota/ Série:** São campos opcionais e servem para o usuário indicar a nota que deu origem à comissão que está sendo lançada. Para lançamentos em que não existe nenhuma nota fiscal ou documento, os campos podem ser deixados em brancos.
- **Observação do lançamento:** Este campo deve ser preenchido com o histórico ou descrição do lançamento de comissão que está sendo efetuado. Tem o objetivo de informar qual a origem do lançamento de débito ou crédito na conta do vendedor.
- **Saldo do Dia:** Este campo é somente para visualização e mostrará o saldo diário de comissão do vendedor informado. (o saldo mostrado será referente a toda movimentação de comissões do vendedor e não somente o saldo dos lançamentos realizados nessa opção).
- **Saldo Atual:** Este campo é somente para visualização e mostrará o saldo atual de comissões do vendedor informado (o saldo mostrado será referente a toda movimentação de comissões do vendedor e não somente o saldo dos lançamentos realizados nessa opção).

Aba de pesquisa

Nesta aba podemos visualizar todos os lançamentos avulsos realizados, no período informado. Se deixado os campos vendedor e nota em branco mostrará todos os lançamentos do período informado.

O usuário poderá consultar, alterar e excluir os lançamentos de comissões nessa aba. Se o lançamento tiver data retroativa e o usuário não tiver permissão para realizar lançamentos com datas retroativas, ele não poderá excluir e nem alterar o lançamento, somente poderá consultar o mesmo.

A Alteração de lançamento de comissão poderá ser realizada através do duplo-clique do mouse ou pelo botão alterar.

A exclusão será feita através do botão excluir da barra de ferramentas ou pressionando ctrl+X.

Obs: A alteração e a exclusão deverão atualizar o saldo de comissões.

Anexo III – User Stories do grupo XP

ID da User Story: **1**	**Tempo implementação:** 5 min.
	Data: 05/03/03
Descrição: Criar uma tela onde será criada uma tabcontrol.	

ID da User Story: **2**	**Tempo implementação:** 10 min.
	Data: 05/03/03
Descrição: Criar uma tabcontrol com duas abas, uma para pesquisa e uma para lançamento.	

ID da User Story: **3**	**Tempo implementação:** 1h e 30 min.
	Data: 05/03/03
Descrição: Colocar na aba de pesquisa os campos para filtros: código da empresa, código do vendedor, data inicial, data final, natureza, valor inicial e valor final.	

ID da User Story: **4**	**Tempo implementação:** 3h e 07 min.
	Data: 06/03/03
Descrição: Mostra menu na barra de ferramentas e implementar as funções dos botões: incluir, excluir, cancelar, imprimir, retornar(fechar tela), salvar.	

ID da User Story: **5**	**Tempo implementação:** 1h e 10 min.
	Data: 06/03/03
Descrição: Criar funções de inclusão e alteração para a aba de lançamento, atualizar as tabelas comissao_movimento e comissao_saldo.	

ID da User Story: **6**	**Tempo implementação:** 1h e 05 min.
	Data: 06/03/03

Descrição: Criar opção de exclusão para a aba de pesquisa, atualizar as tabelas comissao_movimento e comissao_saldo.

ID da User Story: **7**	**Tempo implementação:** 1h e 25 min.
	Data: 06/03/03
Descrição: Criar opção de pesquisa, a partir dos filtros digitados pelo usuário, para a aba de pesquisa.	

ID da User Story: **8**	**Tempo implementação:** 45 min.
	Data: 07/03/03
Descrição: Colocar os campos na aba de lançamento: código da empresa, código do vendedor, nº da nota, série, natureza, valor da base de cálculo, porcentagem da comissão, valor da comissão, data de movimento, tipo movimento, observação, saldo atual.	

ID da User Story: **9**	**Tempo implementação:** 1h e 03 min.
	Data: 07/03/03
Descrição: O código da empresa deverá vir preenchido com a empresa corrente do sistema na aba de lançamento e na aba de pesquisa.	

ID da User Story: **10**	**Tempo implementação:** 54 min.
	Data: 07/03/03
Descrição: A data de movimento na aba de lançamento e a data inicial e data final da aba de pesquisa deverão vir preenhidas com a data atual do sistema operacional da máquina.	

ID da User Story: **11**	**Tempo implementação:** 1h e 15 min.
	Data: 07/03/03

Descrição: Na gravação de um lançamento (inclusão/alteração) os campos obrigatórios deverão estar preechidos, caso não estejam, emitir mensagem para o usuário preencher.

ID da User Story: **12**	**Tempo implementação:** 2h e 45 min.
	Data: 10/03/03
Descrição: Fazer validação para todos os campos da aba de lançamento.	

ID da User Story: **13**	**Tempo implementação:** 1h e 32 min.
	Data: 10/03/03
Descrição: Fazer validação para todos os campos da aba de pesquisa.	

ID da User Story: **14**	**Tempo implementação:** 48 min.
	Data: 11/03/03
Descrição: Fazer implementação para mostrar o saldo do vendedor selecionado no campo Saldo Atual.	

ID da User Story: **15**	**Tempo implementação:** 2h e 38 min.
	Data: 11/03/03
Descrição: A pesquisa deverá mostrar apenas os lançamentos com origem 'A' da tabela comissao_movimento na tela, mais os campos Débito e Crédito do lançamento.	

ID da User Story: **16**	**Tempo implementação:** 2h
	Data: 11/03/03
Descrição: Implementar para que quando o usuário digitar na aba de lançamento o valor da base de cálculo e a porcentagem de comissão, o valor da comissão deverá ser calculado; e se o usuário digitar o valor da base cálculo e o valor da comissão, a porcentagem de comissão deverá ser calculada.	

Anexo IV - Relatório de incidentes de testes

IV.I Incidentes do software da equipe XP

Identificador Incidente	Identificador Espec.	Identificador Caso Teste	Problema
CISSPODER-RT-01-IT-01	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-04	Erro de programa: Botão excluir habilitado quando pesquisa mas não tem lançamentos no banco; tb quando volta da aba de lançamento fica habilitado sem ter pesquisa na tela.
CISSPODER-RT-01-IT-02	CISSPODER-ETF-05	CISSPODER-ETF-05-CT-01	Erro de programa: ao excluir lançamento.
CISSPODER-RT-01-IT-03	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-02	Clicar em uma aba e depois noutra, não permite pois a aba de lançamento por default vem desabilitada, apenas habilitando quando acionar inclusão.
CISSPODER-RT-01-IT-04	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-03	Selecionar uma aba e depois outra (padrão Windows), não permite pois a aba de lançamento por default vem desabilitada, apenas habilitando quando acionar inclusão.
CISSPODER-RT-01-IT-05	CISSPODER-ETF-03	CISSPODER-ETF-03-CT-05	Permite selecionar data inicial maior que data final na aba de pesquisa pelo calendário.
CISSPODER-RT-01-IT-06	CISSPODER-ETF-03	CISSPODER-ETF-03-CT-04	Permite selecionar data inicial maior que data final na aba de pesquisa pelo teclado.
CISSPODER-RT-01-IT-07	CISSPODER-ETF-04	CISSPODER-ETF-04-CT-01	Não permite alterar todos os campos obrigatórios, pois o campo empresa e o campo vendedor ficam desabilitados para o usuário.
CISSPODER-RT-01-IT-08	CISSPODER-ETF-04	CISSPODER-ETF-04-CT-08	Não permite alterar o campo vendedor, pois fica desabilitado para o usuário.
CISSPODER-RT-01-IT-09	CISSPODER-ETF-04	CISSPODER-ETF-04-CT-09	Permite apagar os campos obrigatórios valor base e valor comissão, na alteração de lançamento e gravar.
CISSPODER-RT-01-IT-10	CISSPODER-ETF-04	CISSPODER-ETF-04-CT-11	Permite apagar os campos obrigatórios valor base e valor comissão, na alteração de lançamento e gravar.
CISSPODER-RT-01-IT-11	CISSPODER-ETF-04	CISSPODER-ETF-04-CT-12	Ao gravar inclusão ou alteração de lançamento está mostrando automaticamente a aba de pesquisa com o lançamento incluído/alterado (não deveria).
CISSPODER-RT-01-IT-12	CISSPODER-ETF-05	CISSPODER-ETF-05-CT-04	Não permite fazer pesquisa de lançamentos excluídos, pois a função de exclusão não está funcionando
CISSPODER-RT-01-IT-13	CISSPODER-ETF-05	CISSPODER-ETF-05-CT-03	Não permite fazer pesquisa de lançamentos excluídos, pois a função de exclusão não está funcionando
CISSPODER-RT-01-IT-14	CISSPODER-ETF-06	CISSPODER-ETF-06-CT-03	Inclusão de lançamento preenchendo todos os campos opcionais, permitiu gravar lançamento sem validar campo valor da base e campo valor da comissão (validou apenas a percentagem de comissão).
CISSPODER-RT-01-IT-15	CISSPODER-ETF-06	CISSPODER-ETF-06-CT-04	Inclusão de lançamento preenchendo todos os campos com valores inválidos, permitiu gravar lançamento sem validar campo valor da base, campo percentagem de comissão e campo valor da comissão (foram digitados valores negativos para estes campos), os demais OK.
CISSPODER-RT-01-IT-16	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-01	Não mostrou saldo do vendedor na tela, no momento da inclusão de lançamento com débito.

CISSPODER-RT-01-IT-17	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-02	Não mostrou saldo do vendedor na tela, no momento da inclusão de lançamento com crédito.
CISSPODER-RT-01-IT-18	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-03	Não foi permitido fazer este caso de teste por causa do problema da exclusão de lançamento (não exclui).
CISSPODER-RT-01-IT-19	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-04	Não foi permitido fazer este caso de teste por causa do problema da exclusão de lançamento (não exclui).
CISSPODER-RT-01-IT-20	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-05	Com alteração de lançamento com débito não atualizou o saldo do vendedor.
CISSPODER-RT-01-IT-21	CISSPODER-ETF-07	CISSPODER-ETF-07-CT-06	Com alteração de lançamento com crédito não atualizou o saldo do vendedor.

IV.II Incidentes do software da equipe RUP

Identificador Incidente	Identificador Espec.	Identificador Caso Teste	Problema
CISSPODER-RT-01-IT-01	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-04	Botão incluir habilitado na aba de pesquisa, sendo que deveria ficar desabilitado.Quando o usuário estiver na aba de lançamento deverá ficar habilitado.
CISSPODER-RT-01-IT-02	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-05	Aba lançamento: O botão localizar apenas é habilitado quando o botão cancelar é acionado.
CISSPODER-RT-01-IT-03	CISSPODER-ETF-06	CISSPODER-ETF-06-CT-04	Inclusão de lançamento: Preenchidos todos os campos com valores inválidos e efetuada gravação. Permitiu gravar os campos: valor da base, percentagem de comissão e valor da comissão com valores negativos.
CISSPODER-RT-01-IT-04	CISSPODER-ETF-06	CISSPODER-ETF-06-CT-01	Erro de integridade: ao excluir uma comissão pela tela de lançamento, está limpando a data default do campo data de movimento; e em seguida se fazer uma nova inclusão, no momento da gravação gera o erro de banco.
CISSPODER-RT-01-IT-05	CISSPODER-ETF-01	CISSPODER-ETF-01-CT-04	Botão salvar fica habilitado quando o usuário começa a digitar os filtros para pesquisa.

Anexo V – Scripts de Testes(AutoTeste) da equipe XP

```
of_chama_testes()

boolean lb_Ret
lb_Ret = true

If lb_ret then
     lb_ret = of_existe_campos_dwfiltros()
end if

if lb_ret then
     lb_ret = of_existe_campos_dwlancamento()
end if

If lb_ret then
     lb_ret = of_valida_filtros()
end if

if lb_ret then
     messagebox('Atenção','Teste rodou OK!')
else
     messagebox('Atenção','Teste FALHOU!')
end if

return lb_Ret
```

```
of_existe_campos_dwfiltros()


Integer   li_ctd, li_ctd2, li_cont = 1
String    ls_campos[], ls_colunas[], ls_colatual
Boolean   lb_ret = True, lb_find = False

Datawindow     ldw_filtros
ldw_filtros = tab_comissoes.tabpage_pesquisa.dw_filtros

ls_campos[1] = 'IdEmpresa'
ls_campos[2] = 'IdVendedore'
ls_campos[3] = 'dtinicial'
ls_campos[4] = 'dtfinal'
ls_campos[5] = 'vlrinicial'
ls_campos[6] = 'vlrfinal'
ls_campos[7] = 'natureza'


For li_ctd = 1 to
Integer(ldw_filtros.object.datawindow.column.count)
```

```
     ls_colatual = trim(string(ldw_filtros.Describe("#" +
String(li_ctd) + ".Name")))

     lb_find = False

     For li_ctd2 = 1 to upperbound(ls_campos[])
          If lower(ls_campos[li_ctd2]) = lower(ls_colatual)
then
               lb_find = True
               exit
          end if
     Next

     If lb_find Then
          ls_colunas[li_cont] = ls_colatual
          li_cont = li_cont + 1
     end if
Next


For li_ctd = 1 to upperbound(ls_campos[])

     For li_ctd2 = 1 to upperbound(ls_colunas[])

          If lower(ls_campos[li_ctd]) =
lower(ls_colunas[li_ctd2]) Then
               lb_ret = True
               exit
          else
               lb_ret = False
          end if

     Next

     If not lb_ret then
          Messagebox('Atenção','Coluna '+ls_campos[li_ctd]+'
não existe na DW de Filtros')
          exit
     end if

next

return lb_ret
```

```
of_existe_campos_dwlancamento()

Integer   li_ctd, li_ctd2, li_cont = 1
String    ls_campos[], ls_colunas[], ls_colatual
Boolean   lb_ret = True, lb_find = False
```

```
Datawindow     ldw_lcto
ldw_lcto = tab_comissoes.tabpage_lancamento.dw_lancamento


// Insere as colunas fixas na matriz
ls_campos[1]  = 'IdEmpresa1'
ls_campos[2]  = 'IdVendedore1'
ls_campos[3]  = 'idplanilha'
ls_campos[4]  = 'idclifor1'
ls_campos[5]  = 'dtmovimento'
ls_campos[6]  = 'tiponatureza'
ls_campos[7]  = 'tipomovimento'
ls_campos[8]  = 'numnota'
ls_campos[9]  = 'obslancamento'
ls_campos[10] = 'aliquota'
ls_campos[11] = 'valbasecalculo'
ls_campos[12] = 'valcomissao'
ls_campos[13] = 'serienota'
ls_campos[14] = 'tipoorigempercentual'
ls_campos[15] = 'valsaldoatual'
ls_campos[16] = 'idvendedore'
ls_campos[17] = 'idempresa'
ls_campos[18] = 'idclifore'


For li_ctd = 1 to
Integer(ldw_lcto.object.datawindow.column.count)
     ls_colatual = trim(string(ldw_lcto.Describe("#" +
String(li_ctd) + ".Name")))

     lb_find = False

     For li_ctd2 = 1 to upperbound(ls_campos[])
          If lower(ls_campos[li_ctd2]) = lower(ls_colatual)
then
               lb_find = True
               exit
          end if
     Next

     If lb_find Then
          ls_colunas[li_cont] = ls_colatual
          li_cont = li_cont + 1
     end if
Next


For li_ctd = 1 to upperbound(ls_campos[])
```

```
     For li_ctd2 = 1 to upperbound(ls_colunas[])

          If lower(ls_campos[li_ctd]) =
lower(ls_colunas[li_ctd2]) Then
               lb_ret = True
               exit
          else
               lb_ret = False
          end if

     Next

     If not lb_ret then
          Messagebox('Atenção','Coluna '+ls_campos[li_ctd]+'
não existe na DW de Lançamento')
          exit
     end if

next

return lb_ret
```

```
of_valida_filtros()

boolean lb_ret
lb_ret = true

if
(isnull(tab_comissoes.tabpage_pesquisa.dw_filtros.object.idemp
resa[1])) &
    or
Integer(tab_comissoes.tabpage_pesquisa.dw_filtros.object.idemp
resa[1]) <= 0 then
   lb_Ret = false
     messagebox('Atenção','IdEmpresa com conteúdo inválido')
end if

if lb_ret then
     string ls_dtinicial
     ls_dtinicial =
string(tab_comissoes.tabpage_pesquisa.dw_filtros.object.dtInic
ial[1],"dd/mm/yyyy")
     if not isdate(ls_dtInicial) then
        lb_Ret = false
          messagebox('Atenção','Data Inicial com conteúdo
inválido')
     end if
end if
```

```
if lb_ret then
     string ls_dtfinal
     ls_dtfinal =
string(tab_comissoes.tabpage_pesquisa.dw_filtros.object.dtfina
l[1],"dd/mm/yyyy")
     if not isdate(ls_dtfinal) then
        lb_Ret = false
          messagebox('Atenção','Data Final com conteúdo
inválido')
     end if
end if

if lb_ret then
     if
tab_comissoes.tabpage_pesquisa.dw_filtros.object.dtInicial[1]
> tab_comissoes.tabpage_pesquisa.dw_filtros.object.dtFinal[1]
then
        lb_Ret = false
          messagebox('Atenção','Data Final menor que Data
Inicial')
     end if
end if

if lb_ret then
     if
tab_comissoes.tabpage_pesquisa.dw_filtros.object.vlrInicial[1]
< 0  then
        lb_Ret = false
          messagebox('Atenção','Valor Inicial negativo')
     end if
end if

if lb_ret then
     if
tab_comissoes.tabpage_pesquisa.dw_filtros.object.vlrFinal[1] <
0  then
        lb_Ret = false
          messagebox('Atenção','Valor Final negativo')
     end if
end if

if lb_ret then
     if
tab_comissoes.tabpage_pesquisa.dw_filtros.object.vlrFinal[1] <
tab_comissoes.tabpage_pesquisa.dw_filtros.object.vlrInicial[1]
then
        lb_Ret = false
          messagebox('Atenção','Valor Final menor que Valor
Inicial')
```

```
     end if
end if

return lb_ret
```

Anexo VI – Plano de Testes

Descrição dos Testes do Software

Módulo de Controle de Lançamento de Comissões

Sumário

1. Planos de testes

1.1 Plano de testes de aceitação

1.1.1 Identificador do plano de testes

CISSPODER-PTA-01 – Módulo de Controle de Lançamento de Comissões.

1.1.2 Introdução

1.1.2.1 Objetivos dos testes

Testar a funcionalidade, completeza e correção da implementação do produto CISSPODER – Módulo de Controle de Lançamento Comissão, comparando-o com a respectiva Especificação de Requisitos.

1.1.2.2 Histórico

Um cliente da CISS Automação Comercial especificou dentro do software CISSPODER o desenvolvimento de um módulo para controle e lançamento de comissões. A Especificação de Requisitos do cliente serviu para a implementação do módulo com duas metodologias de desenvolvimento diferentes (eXtreme Programming e Rational Unified Process), visando compara-las na amplitude dos testes com a ocorrências de erros.

1.1.2.3 Escopo dos testes

Todos os Casos de Uso (RUP), User Stories (XP) e requisitos não funcionais levantados na Especificação de Requisitos serão testados.

1.1.3 Itens a testar

Número de ordem	Item	Comentários
	Produto CISSPODER	**O sistema deverá conter a implementação do Controle de Lançamento de Comissões no menu de Controladoria.**

1.1.4 Aspectos a testar

Número de ordem	Item	Referência às Especificações de Testes
	Requisito Selecionar a Opção	**CISSPODER-ETF-01**
	Requisito Informar Vendedor	**CISSPODER-ETF-02**
	Requisito Informar Período	**CISSPODER-ETF-03**
	Requisito Alterar Lançamento de Comissão	**CISSPODER-ETF-04**
	Requisito Excluir Lançamento de Comissão	**CISSPODER-ETF-05**
	Requisito Incluir Lançamento de Comissão	**CISSPODER-ETF-06**
	Requisito Atualizar Saldo de Comissão	**CISSPODER-ETF-07**

1.1.5 Aspectos que não serão testados

Número de ordem	Aspecto	Motivo

1.1.6 Abordagem

Os testes serão feitos de forma manual, executando-se os casos de teste previstos em cada especificação de desenho de teste.

1.1.7 Critérios de completeza e sucesso

Número de ordem	Critério
	Para cada caso de teste, as saídas obtidas são completamente consistentes com as saídas previstas.
	Todas as inclusões, alterações e exclusões de itens foram adequadamente refletidas no estado do banco de dados.
	Todas as operações aritméticas são consistentes.

1.1.8 Critérios de suspensão e retomada

1.1.8.1 Critérios de suspensão dos testes

Ocorrência de Falha Geral de Proteção no Windows ou ruptura do banco de dados.

1.1.8.2 Critérios de retomada dos testes

Correção do código para remoção da causa do erro. Ao ser reiniciado o teste de aceitação, deve ser realizada uma regressão, executando-se novamente todos os casos de teste.

1.1.9 Resultados dos testes

Serão produzidos os seguintes relatórios: **um registro do teste;** **um registro de incidente para cada problema ocorrido;** um relatório resumo do teste.

1.1.10 Tarefas de teste

1.1.11 Ambiente

1.1.11.1 Hardware

Os testes deverão ser executados em um Pentium II 400 Mhz, 128 MB de RAM.

1.1.11.2 Software

O ambiente operacional a ser utilizado é o Windows 98 (ou compatível).

1.1.11.3 Estruturas provisórias de testes

Banco de dados.

1.1.11.4 Documentos

Número de ordem	Documento necessário
	Especificação dos Requisitos do Módulo de Controle de Lançamento Comissões – CISSPODER.
	Descrição dos Testes do Módulo de Controle de Lançamento Comissões – CISSPODER.

1.1.12 Agenda

Vide lista de Tarefas de teste (1.1.10).

1.1.13 Riscos e contingências

Número	**Risco**	**Gravidade**	**Probabilidade de ocorrência**	**Impacto previsto**	**Contramedidas previstas**
1	Falta de povoamento inicial do banco de dados	Alta	Baixa	Impossibilidade de realizar os testes.	Povoar banco de dados com dados relevantes para utilizar nos testes
2	Falta dos equipamentos para os testes.	Alta	Média	Impossibilidade de realizar os testes.	Encontrar equipamentos para substituição.

1.1.14 Aprovação

Aprovamos o documento do Plano de Testes de Aceitação do projeto CISSPODER – Módulo de Controle de Lançamentos de Comissões.

Nome	Organização	Data	Assinatura

1.2 Plano de testes de integração

Não Aplicável

2. Especificações de testes

2.1 Especificação do teste do Requisito Selecionar Opção

2.1.1 Identificador da especificação de teste

CISSPODER-ETF-01

2.1.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Selecionar Opção	**Será testado o acesso às opções de telas.**

2.1.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.1.4 Identificação dos testes

2.1.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
	Pesquisa	**CISSPODER-ETF-01-PT-01**
	Lançamento	**CISSPODER-ETF-01-PT-02**

2.1.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
	Clicar na aba de pesquisa	**CISSPODER-ETF-01-CT-01**
	Clicar na aba de lançamento	**CISSPODER-ETF-01-CT-02**
	Alternar as abas como no padrão Windows	**CISSPODER-ETF-01-CT-03**
	Aba pesquisa, botões habilitados/desabilitados	**CISSPODER-ETF-01-CT-04**
	Aba lançamento, botões habilitados/desabilitados	**CISSPODER-ETF-01-CT-05**

2.1.5 Critérios de completeza e sucesso

Número	Critério
	Clicar na aba de pesquisa para verificar disponibilidade da mesma.
	Clicar na aba de lançamento para verificar disponibilidade da mesma.
	Alternância das abas no padrão Windows, para comprovar utilização do teclado pelo usuário.
	Estando na aba de pesquisa alguns botões da barra de ferramentas devem ficar habilitados e outros desabilitados.
	Estando na aba de lançamento alguns botões da barra de ferramentas devem ficar habilitados e outros desabilitados.

2.1.6 Procedimentos de teste

2.1.6.1 Procedimento de teste Pesquisa

Identificação	CISSPODER-ETF-01-PT-01
Objetivo	Verificar se a aba de pesquisa está disponível ao usuário.
Requisitos especiais	Nenhum
Fluxo	**Selecionar no menu Controladoria a opção Controle de Comissões.** **Clicar na aba de pesquisa, caso abra a tela na aba de lançamento.** **Verificar se todos os campos estão disponíveis.**

2.1.6.3 Procedimento de teste Lançamento

Identificação	CISSPODER-ETF-01-PT-02
Objetivo	Verificar se a aba de lançamento está disponível ao usuário.
Requisitos especiais	Nenhum
Fluxo	**Selecionar no menu Controladoria a opção Controle de Comissões.** **Clicar na aba de lançamento, caso abra a tela na aba de pesquisa.** **Verificar se todos os campos estão disponíveis.**

2.1.7 Casos de teste

2.1.7.1 Caso de teste Clicar na Aba de Pesquisa

Identificação	CISSPODER-ETF-01-CT-01	
Itens a testar	Clicar na aba de pesquisa.	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
		Não aplicável
Ambiente		
Procedimento	Verificar se a aba de pesquisa está disponível ao usuário – CISSPODER-ETF-01-PT-01	
Dependências		

2.1.7.2 Caso de teste Clicar na Aba de Lançamento

Identificação	CISSPODER-ETF-01-CT-02	
Itens a testar	Clicar na aba de lançamento.	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Não aplicável	
Procedimento	Verificar se a aba de lançamento está disponível ao usuário – CISSPODER-ETF-01-PT-02	
Dependências	Nenhuma	

2.1.7.3 Caso de teste Alternar as abas como no padrão Windows

Identificação	CISSPODER-ETF-01-CT-03	
Itens a testar	Alternar as abas como no padrão Windows	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Não aplicável	
Procedimento	Alternar as duas abas como no padrão do Windows, posicionando o foco do cursor em uma das abas e acionar TAB – CISSPODER-ETF-01-PT-01 e CISSPODER-ETF-01-PT-02	
Dependências	Nenhuma	

2.1.7.4 Caso de teste Aba pesquisa, botões habilitados/desabilitados

Identificação	CISSPODER-ETF-01-CT-04	
Itens a testar	Aba pesquisa, botões habilitados/desabilitados	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Não aplicável	
Procedimento	Na aba de pesquisa o botão de cancelar deverá ficar habilitado quando não haver dados na tela. Quando haver dados na tela resultantes de uma pesquisa, os botões de excluir (Ctrl + X), cancelar, imprimir e fechar tela(retornar) deverão ficar habilitados os demais desabilitados – CISSPODER-ETF-01-PT-01	
Dependências	Nenhuma	

2.1.7.5 Caso de teste Aba lançamento, botões habilitados/desabilitados

Identificação	CISSPODER-ETF-01-CT-05	
Itens a testar	Aba lançamento, botões habilitados/desabilitados	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Não aplicável	
Procedimento	No momento de uma inclusão os botões de salvar (Ctrl + S), imprimir, cancelar e fechar tela (retornar) deverão ficar habilitados e os demais desabilitados - CISSPODER-ETF-01-PT-02	
Dependências	Nenhuma	

2.2 Especificação do teste do Requisito Informar Vendedor

2.2.1 Identificador da especificação de teste

CISSPODER-ETF-02

2.2.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Informar Vendedor	**Será testada a funcionalidade do campo do código do vendedor na pesquisa e no lançamento.**

2.2.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.2.4 Identificação dos testes

2.2.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Informar código do vendedor**	**CISSPODER-ETF-02-PT-01**
2	**Informar nome do vendedor**	**CISSPODER-ETF-02-PT-02**

2.2.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Informar código inválido no campo**	**CISSPODER-ETF-02-CT-01**
2	**Informar código válido no campo**	**CISSPODER-ETF-02-CT-02**
3	**Informar nome não cadastrado no campo**	**CISSPODER-ETF-02-CT-03**
4	**Informar nome cadastrado no campo**	**CISSPODER-ETF-02-CT-04**
5	**Informar caracteres inválidos no campo**	**CISSPODER-ETF-02-CT-05**
6	**Selecionar o vendedor pelo botão localizar ao lado do campo**	**CISSPODER-ETF-02-CT-06**

2.2.5 Critérios de completeza e sucesso

Número	Critério
1	**Todos os casos de teste acima testarão o campo código do vendedor que pode receber tanto números(código) como letras(nome) para localização.**
2	**Todos os casos de teste acima testarão a abertura da tela de localização do vendedor.**

2.2.6 Procedimentos de teste

2.2.6.1 Procedimento de teste Informar Código do Vendedor

Identificação	CISSPODER-ETF-02-PT-01
Objetivo	Informar código do vendedor
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de Pesquisa ou Aba de Lançamento.** **Preencher o campo Vendedor com o código.**

2.2.6.2 Procedimento de teste Informar Nome do Vendedor

Identificação	CISSPODER-ETF-02-PT-02
Objetivo	Informar nome do vendedor
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de Pesquisa ou Aba de Lançamento.** **Preencher o campo Vendedor com o nome.**

2.2.7 Casos de teste

2.2.7.1 Caso de teste Informar código inválido no campo

Identificação	CISSPODER-ETF-02-CT-01	
Itens a testar	Informar código inválido no campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	-888.888,00
	Vendedor	9.999.999.999,00
Saídas esperadas	**Campo**	**Valor**
	Mensagem	Vendedor não encontrado
	Mensagem	Vendedor não encontrado
Ambiente	Não aplicável.	
Procedimento	Informar código do vendedor - CISSPODER-ETF-02-PT-01	
Dependências	Nenhuma.	

2.2.7.2 Caso de teste Informar código válido no campo

Identificação	CISSPODER-ETF-02-CT-02	
Itens a testar	Informar código válido no campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	77777
	Vendedor	2
Saídas esperadas	**Campo**	**Valor**
	Mensagem	Vendedor não encontrado
	Vendedor	2
Ambiente	Não aplicável	
Procedimento	Informar código do vendedor - CISSPODER-ETF-02-PT-01	
Dependências	Nenhuma	

2.2.7.3 Caso de teste Informar nome não cadastrado no campo

Identificação	CISSPODER-ETF-02-CT-03	
Itens a testar	Informar nome não cadastrado no campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	François
Saídas esperadas	**Campo**	**Valor**
	Mensagem	Vendedor não encontrado
Ambiente	Não aplicável.	
Procedimento	Informar nome do vendedor - CISSPODER-ETF-02-PT-02	
Dependências	Nenhuma.	

2.2.7.4 Caso de teste Informar nome cadastrado no campo

Identificação	CISSPODER-ETF-02-CT-04	
Itens a testar	Informar nome cadastrado no campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	Vendedor Exemplo
Saídas esperadas	**Campo**	**Valor**
	Vendedor	2
Ambiente	Não aplicável.	
Procedimento	Informar nome do vendedor - CISSPODER-ETF-02-PT-02	
Dependências	Nenhuma.	

2.2.7.5 Caso de teste Informar caracteres inválidos no campo

Identificação	CISSPODER-ETF-02-CT-05	
Itens a testar	Informar caracteres inválidos no campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	%*?!
Saídas esperadas	**Campo**	**Valor**
	Mensagem	Vendedor não encontrado
Ambiente	Não aplicável.	
Procedimento	Informar nome do vendedor - CISSPODER-ETF-02-PT-02	
Dependências	Nenhuma.	

2.2.7.6 Caso de teste Selecionar o vendedor pelo botão localizar ao lado do campo

Identificação	CISSPODER-ETF-02-CT-06	
Itens a testar	Selecionar o vendedor pelo botão localizar ao lado do campo Vendedor	
Entradas	**Campo**	**Valor**
	Botão Localizar	acionar
Saídas esperadas	**Campo**	**Valor**
	Vendedor	Código do vendedor selecionado pelo usuário na tela de localizar.
Ambiente	Não aplicável.	
Procedimento	Informar código do vendedor - CISSPODER-ETF-02-PT-01	
Dependências	Nenhuma.	

2.3 Especificação do teste do Requisito Informar Período

2.3.1 Identificador da especificação de teste

CISSPODER-ETF-03

2.3.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Informar Período	

2.3.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.3.4 Identificação dos testes

2.3.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Informar data pele teclado**	**CISSPODER-ETF-03-PT-01**
2	**Informar data pelo calendário**	**CISSPODER-ETF-03-PT-02**
3	**Informar data inicial maior que data final**	**CISSPODER-ETF-03-PT-03**

2.3.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Informar data inválida pelo teclado**	**CISSPODER-ETF-03-CT-01**
2	Informar data válida pelo teclado	**CISSPODER-ETF-03-CT-02**
3	**Informar data válida pelo calendário**	**CISSPODER-ETF-03-CT-03**
4	Informar data inicial maior que data final pelo calendário	**CISSPODER-ETF-03-CT-04**
5	Informar data inicial maior que data final pelo teclado	**CISSPODER-ETF-03-CT-05**

2.3.5 Critérios de completeza e sucesso

Número	Critério
1	**A data inicial e data final descritas acima fazem referência a aba de pesquisa, onde as mesmas são utilizadas para pesquisar os lançamentos de um determinado período.**

2.3.6 Procedimentos de teste

2.3.6.1 Procedimento de teste Informar data pelo teclado

Identificação	CISSPODER-ETF-03-PT-01
Objetivo	Informar uma data pelo teclado
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de pesquisa ou Aba de lançamento.** **Informar uma data pelo teclado no campo Data.**

2.3.6.2 Procedimento de teste Informar data pelo calendário

Identificação	CISSPODER-ETF-03-PT-02
Objetivo	Informar uma data pelo calendário
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de pesquisa ou Aba de lançamento.** **Informar uma data pelo calendário no campo Data.**

2.3.6.3 Procedimento de teste Informar data inicial maior que data final

Identificação	CISSPODER-ETF-03-PT-03
Objetivo	Informar uma data inicial maior que data final no campo Data
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de pesquisa.** **Informar uma data inicial maior que a data final nos campos Data Inicial e Data Final.**

2.3.7 Casos de teste

2.3.7.1 Caso de teste Informar data inválida pelo teclado no campo

Identificação	CISSPODER-ETF-03-CT-01	
Itens a testar	Informar data inválida pelo teclado no campo Data	
Entradas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	32/12/5555
	Data inicial e data final (Aba pesquisa)	56/21/4856 e 63/57/9663
Saídas esperadas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	Zerar o campo para nova digitação
	Data inicial e data final (Aba pesquisa)	Zerar o campo para nova digitação
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-03-PT-01	
Dependências	Nenhuma.	

2.3.7.2 Caso de teste Informar data válida pelo teclado no campo

Identificação	CISSPODER-ETF-03-CT-02	
Itens a testar	Informar data válida pelo teclado no campo Data	
Entradas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	02/12/2002
	Data inicial e data final (Aba pesquisa)	16/01/2003 e 21/01/2003
Saídas esperadas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	Aceitar a data digitada
	Data inicial e data final (Aba pesquisa)	Aceitar a data digitada
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-03-PT-01	
Dependências	Nenhuma.	

2.3.7.3 Caso de teste Informar data válida pelo calendário

Identificação	CISSPODER-ETF-03-CT-03	
Itens a testar	Informar data válida pelo calendário no campo Data	
Entradas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	02/12/2002
	Data inicial e data final (Aba pesquisa)	16/01/2003 e 21/01/2003
Saídas esperadas	**Campo**	**Valor**
	Data de Movimento (Aba lançamento)	Aceitar a data selecionada e mostrar no campo
	Data inicial e data final (Aba pesquisa)	Aceitar a data selecionada e mostrar nos campos
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-03-PT-02	
Dependências	Nenhuma.	

2.3.7.4 Caso de teste Informar data inicial maior que data final (aba pesquisa) pelo teclado

Identificação	CISSPODER-ETF-03-CT-04	
Itens a testar	Informar data inicial maior que data final (aba pesquisa) pelo teclado no campo Data	
Entradas	**Campo**	**Valor**
	Data inicial	23/02/2003
	Data final	16/02/2003
Saídas esperadas	**Campo**	**Valor**
	Data inicial	Zerar o campo para nova digitação
	Data final	Zerar o campo para nova digitação
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-03-PT-03	
Dependências	Nenhuma.	

2.3.7.5 Caso de teste Informar data inicial maior que data final (aba pesquisa) pelo calendário

Identificação	CISSPODER-ETF-03-CT-05	
Itens a testar	Informar data inicial maior que data final (aba pesquisa) pelo calendário no campo Data	
Entradas	**Campo**	**Valor**
	Data inicial	Selecionar 23/02/2003
	Data final	Selecionar 16/02/2003
Saídas esperadas	**Campo**	**Valor**
	Data inicial	Zerar o campo para nova seleção
	Data final	Zerar o campo para nova seleção
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-03-PT-03	
Dependências	Nenhuma.	

2.4 Especificação do teste do Requisito Alterar lançamento comissão

2.4.1 Identificador da especificação de teste

CISSPODER-ETF-04

2.4.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Alterar Lançamento Comissão	

2.4.3 Detalhes da abordagem

Apenas na Aba de lançamento as alterações dos lançamentos serão feitas. Os dados dos lançamentos são levados para a tela e colocados em seus respectivos campos, como no momento da gravação. Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.4.4 Identificação dos testes

2.4.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Alterar campos obrigatórios**	**CISSPODER-ETF-04-PT-01**
2	**Alterar campos opcionais**	**CISSPODER-ETF-04-PT-02**
3	**Pesquisar lançamento alterado**	**CISSPODER-ETF-04-PT-03**

2.4.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Alterar todos os campos obrigatórios**	**CISSPODER-ETF-04-CT-01**
2	**Alterar todos os campos opcionais**	**CISSPODER-ETF-04-CT-02**
3	**Alterar todos os campos (obrigatórios e opcionais)**	**CISSPODER-ETF-04-CT-03**
4	**Alterar o campo Valor da Base**	**CISSPODER-ETF-04-CT-04**
5	**Alterar o campo % Comissão**	**CISSPODER-ETF-04-CT-05**
6	**Alterar o campo Valor da Comissão**	**CISSPODER-ETF-04-CT-06**
7	**Alterar o campo Natureza**	**CISSPODER-ETF-04-CT-07**
8	**Alterar o campo Vendedor**	**CISSPODER-ETF-04-CT-08**
9	**Apagar todos os campos obrigatórios**	**CISSPODER-ETF-04-CT-09**
10	**Apagar todos os campos opcionais**	**CISSPODER-ETF-04-CT-10**
11	**Apagar todos os campos (obrigatórios e opcionais)**	**CISSPODER-ETF-04-CT-11**
12	**Pesquisar lançamento alterado**	**CISSPODER-ETF-04-CT-12**

2.4.5 Critérios de completeza e sucesso

Número	Critério
1	**Todos os lançamentos alterados são localizados com seus dados gravados na Aba de pesquisa.**
2	**Todos os lançamentos alterados são atualizados na tabela comissao_movimento e comissao_saldo.**

2.4.6 Procedimentos de teste

2.4.6.1 Procedimento de teste Alterar campos obrigatórios

Identificação	CISSPODER-ETF-04-PT-01
Objetivo	Alterar campos obrigatórios
Requisitos especiais	Os campos obrigatórios devem ser alterados.
Fluxo	**No Menu Controladoria escolher a opção Controle de Comissões.** **Selecionar a Aba de Pesquisa.** **Pesquisar um lançamento.** **Selecioná-lo.** **Acionar o botão Alterar ou duplo-clique sobre o lançamento.** **Alterar os campos obrigatórios.**

2.4.6.2 Procedimento de teste Alterar campos opcionais

Identificação	CISSPODER-ETF-04-PT-02
Objetivo	Alterar campos opcionais
Requisitos especiais	O campos opcionais devem ser alterados.
Fluxo	**No Menu Controladoria escolher a opção Controle de Comissões.** **Selecionar a Aba de Pesquisa.** **Pesquisar um lançamento.** **Selecioná-lo.** **Acionar o botão Alterar ou duplo-clique sobre o lançamento.** **Alterar os campos opcionais.**

2.4.6.3 Procedimento de teste Pesquisar lançamento alterado

Identificação	CISSPODER-ETF-04-PT-03
Objetivo	Pesquisar lançamento alterado
Requisitos especiais	O campos opcionais devem ser alterados.
Fluxo	**No Menu Controladoria escolher a opção Controle de Comissões.** **Selecionar a Aba de Pesquisa.** **Pesquisar o lançamento alterado.** **Verificar se os campos alterados conferem.**

2.4.7 Casos de teste

2.4.7.1 Caso de teste Alterar todos os campos obrigatórios

Identificação	CISSPODER-ETF-04-CT-01	
Itens a testar	Alterar todos os campos obrigatórios	
Entradas	**Campo**	**Valor**
	Natureza	Crédito/Débito
	Valor Base	400,00
	Porcentagem Comissão	10
	Valor Comissão	40,00
	Data Movimento	Data atual
	Vendedor	2
Saídas esperadas	**Campo**	**Valor**
	Natureza	Crédito/Débito
	Valor Base	400,00
	Porcentagem Comissão	10
	Valor Comissão	40
	Data Movimento	Data atual
	Vendedor	2
Ambiente	Banco de dados inicial	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.2 Caso de teste Alterar todos os campos opcionais

Identificação	CISSPODER-ETF-04-CT-02	
Itens a testar	Alterar todos os campos opcionais	
Entradas	**Campo**	**Valor**
	Cliente/fornecedor	27
	Nº nota	4656
	Série	1
	Observação	Teste ETF04-CT02
Saídas esperadas	**Campo**	**Valor**
	Cliente/fornecedor	27
	Nº nota	4656
	Série	1
	Observação	Teste ETF04-CT02
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-02	
Dependências	Nenhuma.	

2.4.7.3 Caso de teste Alterar todos os campos (obrigatórios e opcionais)

Identificação	CISSPODER-ETF-04-CT-03	
Itens a testar	Alterar todos os campos (obrigatórios e opcionais)	
Entradas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	400,00
	Porcentagem de comissão	10
	Valor da comissão	40,00
	Cliente/fornecedor	27
	Nº nota	566
	Série	1
	Observação	Teste ETF04-CT3
Saídas esperadas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	400,00
	Porcentagem de comissão	10
	Valor da comissão	40,00
	Cliente/fornecedor	27
	Nº nota	566
	Série	1
	Observação	Teste ETF04-CT3
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01 e CISSPODER-ETF-04-PT-02	
Dependências	Nenhuma.	

2.4.7.4 Caso de teste Alterar o campo Valor da Base

Identificação	CISSPODER-ETF-04-CT-04	
Itens a testar	Alterar o campo Valor da Base	
Entradas	**Campo**	**Valor**
	Valor da base de cálculo	550,00
Saídas esperadas	**Campo**	**Valor**
	Valor da base de cálculo	550,00
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.5 Caso de teste Alterar o campo Porcentagem Comissão

Identificação	CISSPODER-ETF-04-CT-05	
Itens a testar	Alterar o campo Porcentagem Comissão	
Entradas	**Campo**	**Valor**
	Porcentagem Comissão	15
Saídas esperadas	**Campo**	**Valor**
	Porcentagem Comissão	15
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.6 Caso de teste Alterar o campo Valor da Comissão

Identificação	CISSPODER-ETF-04-CT-06	
Itens a testar	Alterar o campo Valor da Comissão	
Entradas	**Campo**	**Valor**
	Valor da Comissão	50,00
Saídas esperadas	**Campo**	**Valor**
	Valor da Comissão	50,00
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.7 Caso de teste Alterar o campo Natureza

Identificação	CISSPODER-ETF-04-CT-07	
Itens a testar	Alterar o campo Natureza	
Entradas	**Campo**	**Valor**
	Natureza	Débito/Crédito – Crédito/Débito
Saídas esperadas	**Campo**	**Valor**
	Natureza	Débito/Crédito – Crédito/Débito
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.8 Caso de teste Alterar o campo Vendedor

Identificação	CISSPODER-ETF-04-CT-08	
Itens a testar	Alterar o campo Vendedor	
Entradas	**Campo**	**Valor**
	Vendedor	2
Saídas esperadas	**Campo**	**Valor**
	Vendedor	2
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.9 Caso de teste Apagar todos os campos obrigatórios

Identificação	CISSPODER-ETF-04-CT-09	
Itens a testar	Apagar todos os campos obrigatórios	
Entradas	**Campo**	**Valor**
	Natureza	Crédito/Débito – Débito/Crédito
	Valor Base	{em branco}
	Porcentagem Comissão	{em branco}
	Valor Comissão	{em branco}
	Data Movimento	00/00/0000
	Vendedor	{em branco}
Saídas esperadas	**Campo**	**Valor**
	Natureza	Crédito/Débito – Débito/Crédito
	Valor Base	Mensagem: preencher o campo valor da base
	Porcentagem Comissão	Mensagem: preencher o campo porcentagem comissão
	Valor Comissão	Mensagem: preencher o campo valor comissão
	Data Movimento	Mensagem: preencher o campo data movimento
	Vendedor	Vendedor não encontrado.
Ambiente	Banco de dados inicial	
Procedimento	CISSPODER-ETF-04-PT-01	
Dependências	Nenhuma.	

2.4.7.10 Caso de teste Apagar todos os campos opcionais

Identificação	CISSPODER-ETF-04-CT-10	
Itens a testar	Apagar todos os campos opcionais	
Entradas	**Campo**	**Valor**
	Cliente/fornecedor	{em branco}
	Nº nota	{em branco}
	Série	{em branco}
	Observação	{em branco}
Saídas esperadas	**Campo**	**Valor**
	Cliente/fornecedor	{em branco}
	Nº nota	{em branco}
	Série	{em branco}
	Observação	{em branco}
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-02	
Dependências	Nenhuma.	

2.4.7.11 Caso de teste Apagar todos os campos (obrigatórios e opcionais)

Identificação	CISSPODER-ETF-04-CT-11	
Itens a testar	Apagar todos os campos (obrigatórios e opcionais)	
Entradas	**Campo**	**Valor**
	Vendedor	{em branco}
	Data movimento	{em branco}
	Natureza	Débito/Crédito – Crédito/Débito
	Valor base cálculo	{em branco}
	Porcentagem de comissão	{em branco}
	Valor da comissão	{em branco}
	Cliente/fornecedor	{em branco}
	Nº nota	{em branco}
	Série	{em branco}
	Observação	{em branco}
Saídas esperadas	**Campo**	**Valor**
	Vendedor	Mensagem: preencher o campo vendedor
	Data movimento	Mensagem: preencher o campo data movimento
	Natureza	Débito/Crédito – Crédito/Débito
	Valor base cálculo	Mensagem: preencher o campo valor base cálculo
	Porcentagem de comissão	Mensagem: preencher o campo porcentagem comissão
	Valor da comissão	Mensagem: preencher o campo valor comissão
	Cliente/fornecedor	{em branco}
	Nº nota	{em branco}
	Série	{em branco}
	Observação	{em branco}
Ambiente	Não aplicável	
Procedimento	CISSPODER-ETF-04-PT-01 e CISSPODER-ETF-04-PT-02	
Dependências	Nenhuma.	

2.4.7.12 Caso de teste Pesquisar lançamento alterado

Identificação	CISSPODER-ETF-04-CT-12	
Itens a testar	Pesquisar lançamento alterado	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado.	
Procedimento	CISSPODER-ETF-04-PT-03	
Dependências	Este caso de teste será executado toda vez que o lançamento for alterado, para verificar se ocorreu a alteração.	

2.5 Especificação do teste do Requisito Excluir lançamento de comissão

2.5.1 Identificador da especificação de teste

CISSPODER-ETF-05

2.5.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Excluir lançamento de comissão	

2.5.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.5.4 Identificação dos testes

2.5.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Exclusão de lançamento**	**CISSPODER-ETF-05-PT-01**
2	**Pesquisa de exclusão de lançamento**	**CISSPODER-ETF-05-PT-02**

2.5.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Excluir lançamento pelo botão excluir**	**CISSPODER-ETF-05-CT-01**
2	Excluir lançamento pelo atalho Ctrl+X	**CISSPODER-ETF-05-CT-02**
3	**Pesquisar lançamento informando todos os filtros da pesquisa**	**CISSPODER-ETF-05-CT-03**
4	**Pesquisar lançamento informando Empresa e Vendedor**	**CISSPODER-ETF-05-CT-04**

2.5.5 Critérios de completeza e sucesso

Número	Critério
1	**A exclusão de um lançamento de comissão é feito pela tela de pesquisa exclusivamente; para cada exclusão a pesquisa deve ser acionada para verificar se o lançamento foi excluído do banco de dados.**

2.5.6 Procedimentos de teste

2.5.6.1 Procedimento de teste Exclusão de lançamento

Identificação	CISSPODER-ETF-05-PT-01
Objetivo	Exclusão de lançamento
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de pesquisa.** **Pesquisar o lançamento a ser excluído.** **Selecionar o lançamento.** **5. Acionar o botão Excluir da barra de ferramentas.**

2.5.6.2 Procedimento de teste Pesquisa de exclusão de lançamento

Identificação	CISSPODER-ETF-05-PT-02
Objetivo	Pesquisa de exclusão de lançamento
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria escolher a opção Controle de Comissões.** **Selecionar a Aba de pesquisa.** **3. Pesquisar o lançamento excluído.**

2.5.7 Casos de teste

2.5.7.1 Caso de teste Excluir lançamento pelo botão excluir

Identificação	CISSPODER-ETF-05-CT-01	
Itens a testar	Excluir lançamento pelo botão excluir	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-05-PT-01	
Dependências	Para execução deste caso de teste o lançamento a ser excluído deve estar selecionado.	

2.5.7.2 Caso de teste Excluir lançamento pelo atalho Ctrl+X

Identificação	CISSPODER-ETF-05-CT-02	
Itens a testar	Excluir lançamento pelo atalho Ctrl+X	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-05-PT-01	
Dependências	Para execução deste caso de teste o lançamento a ser excluído deve estar selecionado.	

2.5.7.3 Caso de teste Pesquisar lançamento informando todos os filtros da pesquisa

Identificação	CISSPODER-ETF-05-CT-03	
Itens a testar	Pesquisar lançamento informando todos os filtros da pesquisa	
Entradas	**Campo**	**Valor**
	Empresa	1
	Vendedor	2
	Data inicial	02/03/2003
	Data final	29/03/2003
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-05-PT-02	
Dependências	Para execução deste caso de teste, a saída esperada é a pesquisa mostrada na tela, ela será utilizada para verificar se o lançamento foi excluído.	

2.5.7.4 Caso de teste Pesquisar lançamento informando Empresa e Vendedor

Identificação	CISSPODER-ETF-05-CT-04	
Itens a testar	Pesquisar lançamento informando Empresa e Vendedor	
Entradas	**Campo**	**Valor**
	Empresa	1
	Vendedor	2
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-05-PT-02	
Dependências	Para execução deste caso de teste, a saída esperada é a pesquisa mostrada na tela, ela será utilizada para verificar se o lançamento foi excluído.	

2.6 Especificação do teste do Requisito Incluir lançamento de comissão

2.6.1 Identificador da especificação de teste

CISSPODER-ETF-06

2.6.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Incluir lançamento de comissão	

2.6.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.6.4 Identificação dos testes

2.6.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Inclusão de lançamento**	**CISSPODER-ETF-06-PT-01**
2	**Pesquisa de inclusão de lançamento**	**CISSPODER-ETF-06-PT-02**

2.6.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Incluir lançamento preenchendo todos os campos**	**CISSPODER-ETF-06-CT-01**
2	Incluir lançamento preenchendo todos os campos obrigatórios	**CISSPODER-ETF-06-CT-02**
3	**Incluir lançamento preenchendo todos os campos opcionais**	**CISSPODER-ETF-06-CT-03**
4	**Incluir lançamento preenchendo todos os campos com valores inválidos**	**CISSPODER-ETF-06-CT-04**
5	**Pesquisar lançamento informando todos os filtros da pesquisa**	**CISSPODER-ETF-06-CT-05**
6	**Pesquisar lançamento informando Empresa e Vendedor**	**CISSPODER-ETF-06-CT-06**

2.6.5 Critérios de completeza e sucesso

Número	Critério
1	**A inclusão de um lançamento de comissão é feito pela tela de lançamento exclusivamente; para cada inclusão a pesquisa deve ser acionada para verificar se o lançamento foi incluído no banco de dados.**

2.6.6 Procedimentos de teste

2.6.6.1 Procedimento de teste Inclusão de lançamento

Identificação	CISSPODER-ETF-06-PT-01
Objetivo	Inclusão de lançamento
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de lançamento.** **Preencher os campos.** **4. Acionar gravação do lançamento.**

2.6.6.2 Procedimento de teste Pesquisa de inclusão de lançamento

Identificação	CISSPODER-ETF-06-PT-02
Objetivo	Pesquisa de inclusão de lançamento
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de pesquisa.** **3. Pesquisar o lançamento incluído no banco de dados.**

2.6.7 Casos de teste

2.6.7.1 Caso de teste Incluir lançamento preenchendo todos os campos

Identificação	CISSPODER-ETF-06-CT-01	
Itens a testar	Incluir lançamento preenchendo todos os campos	
Entradas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	30,00
	Porcentagem de comissão	10
	Valor da comissão	3,00
	Cliente/fornecedor	27
	Nº nota	5688
	Série	2
	Observação	Teste ETF06-CT01
Saídas esperadas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	30,00
	Porcentagem de comissão	10
	Valor da comissão	3,00
	Cliente/fornecedor	27
	Nº nota	5688
	Série	2
	Observação	Teste ETF06-CT01
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-06-PT-01	
Dependências	Nenhuma.	

2.6.7.2 Caso de teste Incluir lançamento preenchendo todos os campos obrigatórios

Identificação	CISSPODER-ETF-06-CT-02	
Itens a testar	Incluir lançamento preenchendo todos os campos obrigatórios	
Entradas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	300,00
	Porcentagem de comissão	10
	Valor da comissão	30,00
Saídas esperadas	**Campo**	**Valor**
	Vendedor	2
	Data movimento	Data atual
	Natureza	Débito/Crédito
	Valor base cálculo	300,00
	Porcentagem de comissão	10
	Valor da comissão	30,00
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-06-PT-01	
Dependências	Nenhuma.	

2.6.7.3 Caso de teste Incluir lançamento preenchendo todos os campos opcionais

Identificação	CISSPODER-ETF-06-CT-03	
Itens a testar	Incluir lançamento preenchendo todos os campos opcionais	
Entradas	**Campo**	**Valor**
	Cliente/Fornecedor	27
	Nº nota	455
	Série	2
	Observação	Teste ETF06-CT03
Saídas esperadas	**Campo**	**Valor**
	Cliente/Fornecedor	27
	Nº nota	455
	Série	2
	Observação	Teste ETF06-CT03
	Mensagem:	Preencher campos obrigatórios antes de efetuar gravação
Ambiente	Banco de dados utilizado.	
Procedimento	CISSPODER-ETF-06-PT-01	
Dependências	Nenhuma.	

2.6.7.4 Caso de teste Incluir lançamento preenchendo todos os campos com valores inválidos

Identificação	CISSPODER-ETF-06-CT-04	
Itens a testar	Incluir lançamento preenchendo todos os campos com valores inválidos	
Entradas	**Campo**	**Valor**
	Vendedor	&%*
	Data movimento	65/15/2005
	Natureza	Débito/Crédito
	Valor base cálculo	-30,00
	Porcentagem de comissão	-10
	Valor da comissão	-3,00
	Cliente/fornecedor	+/@
	Nº nota	Nota Nota
	Série	Serie serie
	Observação	) (%%
Saídas esperadas	**Campo**	**Valor**
	Vendedor	Vendedor não encontrado.
	Data movimento	Não deve aceitar a data informada.
	Natureza	Débito/Crédito
	Valor base cálculo	Mensagem: valor inválido para o campo
	Porcentagem de comissão	Mensagem: valor inválido para o campo
	Valor da comissão	Mensagem: valor inválido para o campo
	Cliente/fornecedor	Cliente/fornecedor não encontrado.
	Nº nota	Mensagem: valor inválido para o campo
	Série	Mensagem: valor inválido para o campo
	Observação	) (%%
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-06-PT-01	
Dependências	Nenhuma.	

2.6.7.5 Caso de teste Pesquisar lançamento informando todos os filtros da pesquisa

Identificação	CISSPODER-ETF-06-CT-05	
Itens a testar	Pesquisar lançamento informando todos os filtros da pesquisa	
Entradas	**Campo**	**Valor**
	Empresa	1
	Vendedor	2
	Data inicial	02/03/2003
	Data final	29/03/2003
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-06-PT-02	
Dependências	Para execução deste caso de teste, a saída esperada é a pesquisa mostrada na tela, ela será utilizada para verificar se o lançamento foi incluído no banco de dados.	

2.6.7.6 Caso de teste Pesquisar lançamento informando Empresa e Vendedor

Identificação	CISSPODER-ETF-06-CT-06	
Itens a testar	Pesquisar lançamento informando Empresa e Vendedor	
Entradas	**Campo**	**Valor**
	Empresa	1
	Vendedor	2
Saídas esperadas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-06-PT-02	
Dependências	Para execução deste caso de teste, a saída esperada é a pesquisa mostrada na tela, ela será utilizada para verificar se o lançamento foi incluído no banco de dados.	

2.7 Especificação do teste do Requisito Atualizar saldo de comissão

2.7.1 Identificador da especificação de teste

CISSPODER-ETF-07

2.7.2 Aspectos a serem testados

Número	Requisito	Comentários
	Requisito Atualizar saldo de comissão	

2.7.3 Detalhes da abordagem

Os casos de teste deverão ser executados na ordem em que são aqui apresentados.

2.7.4 Identificação dos testes

2.7.4.1 Procedimentos de teste

Número	Procedimento de teste	Identificação do procedimento de teste
1	**Atualizar saldo com débito**	**CISSPODER-ETF-07-PT-01**
2	**Atualizar saldo com crédito**	**CISSPODER-ETF-07-PT-02**

2.7.4.2 Casos de teste

Número	Caso de teste	Identificação do caso de teste
1	**Lançamento de comissão com débito**	**CISSPODER-ETF-07-CT-01**
2	Lançamento de comissão com crédito	**CISSPODER-ETF-07-CT-02**
3	**Exclusão de lançamento com débito**	**CISSPODER-ETF-07-CT-03**
4	**Exclusão de lançamento com crédito**	**CISSPODER-ETF-07-CT-04**
5	**Alteração de lançamento com débito**	**CISSPODER-ETF-07-CT-05**
6	**Alteração de lançamento com crédito**	**CISSPODER-ETF-07-CT-06**

2.7.5 Critérios de completeza e sucesso

Número	Critério
1	**A atualização do saldo de comissões faz-se necessariamente através de uma inclusão, uma alteração ou uma exclusão de lançamento do banco de dados.**

2.7.6 Procedimentos de teste

2.7.6.1 Procedimento de teste Atualizar saldo com débito

Identificação	CISSPODER-ETF-07-PT-01
Objetivo	Atualizar saldo com débito
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de lançamento -** Incluir lançamento**.** **Selecionar a Aba de pesquisa -** alterar ou excluir lançamento**.** **Preencher os campos aba lançamento, com Natureza(D) –** incluir. **Preencher os filtros aba pesquisa -** alterar ou excluir**.** **Acionar gravação aba lançamento –** incluir. **Acionar pesquisa aba pesquisa -** alterar ou excluir**.** **Selecionar lançamento aba pesquisa -** alterar ou excluir**.** **Aba pesquisa acionar botão Excluir –** excluir. **Duplo clique -** alterar**.** **Aba lançamento, alterar o campo Natureza(D) necessário para alterar saldo com débito, acionar gravação -** alterar**.**

2.7.6.2 Procedimento de teste Atualizar saldo com crédito

Identificação	CISSPODER-ETF-07-PT-02
Objetivo	Atualizar saldo com crédito
Requisitos especiais	Nenhum
Fluxo	**Selecionar no Menu Controladoria a opção Controle de Comissões.** **Selecionar a Aba de lançamento -** Incluir lançamento**.** **Selecionar a Aba de pesquisa -** alterar ou excluir lançamento**.** **Preencher os campos aba lançamento, com Natureza(C) –** incluir. **Preencher os filtros aba pesquisa -** alterar ou excluir**.** **Acionar gravação aba lançamento –** incluir. **Acionar pesquisa aba pesquisa -** alterar ou excluir**.** **Selecionar lançamento aba pesquisa -** alterar ou excluir**.** **Aba pesquisa acionar botão Excluir –** excluir. **Duplo clique -** alterar**.** **11. Aba lançamento, alterar o campo Natureza(C) necessário para alterar saldo com crédito, acionar gravação -** alterar**.**

2.7.7 Casos de Teste

2.7.7.1 Caso de teste Lançamento de comissão com débito

Identificação	CISSPODER-ETF-07-CT-01	
Itens a testar	Lançamento de comissão com débito	
Entradas	**Campo**	**Valor**
	Natureza	Débito
	Valor base	600,00
	Porcentagem Comissão	10
	Valor Comissão	60,00
Saídas esperadas	**Campo**	**Valor**
	Natureza	Débito
	Valor base	600,00
	Porcentagem Comissão	10
	Valor Comissão	60,00
	Saldo atual	Saldo atual anterior menos 60,00
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-01	
Dependências	Nenhuma.	

2.7.7.2 Caso de teste Lançamento de comissão com crédito

Identificação	CISSPODER-ETF-07-CT-02	
Itens a testar	Lançamento de comissão com crédito	
Entradas	**Campo**	**Valor**
	Natureza	Crédito
	Valor base	600,00
	Porcentagem Comissão	10
	Valor Comissão	60,00
Saídas esperadas	**Campo**	**Valor**
	Natureza	Crédito
	Valor base	600,00
	Porcentagem Comissão	10
	Valor Comissão	60,00
	Saldo atual	Saldo atual anterior mais 60,00
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-02	
Dependências	Nenhuma.	

2.7.7.3 Caso de teste Exclusão de lançamento débito

Identificação	CISSPODER-ETF-07-CT-03	
Itens a testar	Exclusão de lançamento com débito	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Saldo atual	Saldo atual anterior menos valor débito do lançamento excluído
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-01	
Dependências	Este caso de teste deve ser executado para um lançamento com valor débito; atualizando assim a tabela comissao_saldo.	

2.7.7.4 Caso de teste Exclusão de lançamento crédito

Identificação	CISSPODER-ETF-07-CT-04	
Itens a testar	Exclusão de lançamento com crédito	
Entradas	**Campo**	**Valor**
	Não aplicável	Não aplicável
Saídas esperadas	**Campo**	**Valor**
	Saldo atual	Saldo atual anterior menos valor crédito do lançamento excluído
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-02	
Dependências	Este caso de teste deve ser executado para um lançamento com valor crédito; atualizando assim a tabela comissao_saldo.	

2.7.7.5 Caso de teste Alteração de lançamento débito

Identificação	CISSPODER-ETF-07-CT-05	
Itens a testar	Alteração de lançamento com débito	
Entradas	**Campo**	**Valor**
	Natureza	Débito
	Valor base	700,00
	Porcentagem comissão	10
	Valor comissão	70,00
Saídas esperadas	**Campo**	**Valor**
	Natureza	Débito
	Valor base	700,00
	Porcentagem comissão	10
	Valor comissão	70,00
	Saldo atual	Saldo atual anterior menos 70,00
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-01	
Dependências	Este caso de teste deve ser executado para um lançamento com valor débito; atualizando assim a tabela comissao_saldo.	

2.7.7.6 Caso de teste Alteração de lançamento crédito

Identificação	CISSPODER-ETF-07-CT-06	
Itens a testar	Alteração de lançamento com crédito	
Entradas	**Campo**	**Valor**
	Natureza	Crédito
	Valor base	700,00
	Porcentagem comissão	10
	Valor comissão	70,00
Saídas esperadas	**Campo**	**Valor**
	Natureza	Crédito
	Valor base	700,00
	Porcentagem comissão	10
	Valor comissão	70,00
	Saldo atual	Saldo atual anterior mais 70,00
Ambiente	Banco de dados utilizado	
Procedimento	CISSPODER-ETF-07-PT-02	
Dependências	Este caso de teste deve ser executado para um lançamento com valor crédito; atualizando assim a tabela comissao_saldo.	