UNIVERSIDADE DO EXTEMO SUL CATARINENSE – UNESC

CURSO DE CIENCIA DA COMPUTAÇÃO

TIAGO CLARINDA

O GERENCIAMENTO DE RISCOS APLICADO NA ENGENHARIA DE REQUISITOS PARA EVITAR RISCOS NO DESENVOLVIMENTO DE SOFTWARE: ESTUDO DE CASO

CRICIÚMA, JULHO DE 2010

TIAGO CLARINDA

O GERENCIAMENTO DE RISCOS APLICADO NA ENGENHARIA DE REQUISITOS PARA EVITAR RISCOS NO DESENVOLVIMENTO DE SOFTWARE: ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MSc. Paracelso de Oliveira Caldas

CRICIUMA, JULHO DE 2010

TIAGO CLARINDA

O GERENCIAMENTO DE RISCOS APLICADO NA ENGENHARIA DE REQUISITOS PARA EVITAR RISCOS NO DESENVOLVIMENTO DE SOFTWARE: ESTUDO DE CASO

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paracelso de Oliveira Caldas (UNESC)
Orientador

Prof. MSc. Gustavo Bisognin (UNESC)

Prof. Esp. Luciano Antunes (UNESC)

A minha madrinha, mãe e avó, Ana Maria, Sandra e Terezinha, sem eles, nada disso seria possível.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus que me deu força e coragem para realizar esse trabalho. Agradeço por sempre colocar pessoas certas em meu caminho, que sempre me auxiliaram, não só nesse trabalho, como em toda a vida.

Agradeço a minha família, por todo amor, carinho, compreensão e em especial a minha madrinha por ter acreditado e investido na minha formação, me abrindo as portas para minha formação profissional. A minha avó materna Dona Terezinha agradeço por todas as noites em que ficava a minha espera me incentivando e apoiando, após as aulas, toda a saúde desse mundo para senhora. E a minha mãe pela cobrança e estimulo durante o curso.

A todos os meus colegas de graduação, por compartilharem suas experiências de vidas e momentos de descontração e diversão durante todo o curso.

Ao meu Orientador, Prof. MSc. Paracelso de Oliveira Caldas por acreditar no meu projeto, em minha capacidade e por toda atenção e ajuda prestada durante todo o andamento desta pesquisa.

Enfim, a todos que contribuíram para a execução desse trabalho, seja pela ajuda constante ou por uma simples palavras de afeto.

RESUMO

A presente pesquisa refere-se à análise e aplicação do gerenciamento de riscos na fase de Engenharia de Requisitos do ciclo de vida de um software para identificar e evitar riscos. Para tal finalidade, foram evidenciados conceitos da Engenharia de Software referentes a processos, métodos, normas e padrões de desenvolvimento de software e Engenharia de Requisitos. Esta fase proporciona vários riscos ao desenvolvimento do software e gerenciá-los se mostra um método eficaz de combate a problemas futuros. Conceitos de gerenciamento de riscos foram abordados e aplicados a um Projeto de Software sendo analisada as etapas que produziram os requisitos. O intuito foi identificar pontos vulneráveis do documento de requisitos e produzir planos de ação para evitá-los. Com isso, foram evidenciados diversos riscos nos requisitos e seus processos formando um escopo que serve de base para identificação de riscos em outros projetos. O combate a riscos de requisitos logo no inicio do projeto além de baixar os custos vai evitar que os riscos se transformem em problemas e se propaguem originando outros para o projeto. A produção dos requisitos de forma incorreta, como a analisada, formará um software desastroso.

Palavras-chave: Engenharia de Software; Gerenciamento de Riscos; Engenharia de Requisitos; Requisitos.

ABSTRACT

The present study concerns the analysis and application of risk management in the Requirement Engineering phase of a software life cycle to identify and avoid risks. For this purpose, concepts of Software Engineering regarding processes, methods, norms and standards for software development and Requirement Engineering were highlighted. This phase provides several risks to software development and manage them proves to be an effective method to combat future problems. Concepts of risk management were discussed and applied to a Software Project by reviewing the steps that produced the requirements. The aim was to identify vulnerable points in the requirements document and produce action plans to avoid them. Thus, several risks were highlighted in the requirements and in their processes, forming a scope that is the basis for identifying risks in other projects. The combat of risks requirements in the beginning of a project in addition to lowering costs will prevent risks from becoming problems and propagating themselves causing more problems for the project. The production of requirements incorrectly, as discussed, will form a disastrous software.

Keywords: Software Engineering, Risk Management, Requirements Engineering, Requirements.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

CMMI	*Capability Maturity Model Integrated*
ER	Engenharia de Requisitos
ES	Engenharia de *Software*
GR	Gerência de Riscos
IEEE	*Institute of Electrical and Electronics Engineers*
ISO	*International Organization for Standardization*
PMBOK	*Project Management Body of Knowledge*
PS	Processo de *Software*
RUP	*Rational Unified Process*
SDCE	*Software Development Capability Evaluation*
SEI	*Software Engineering Institute*
SMPVRV	Sistema de Manutenção de Pedidos de Venda de Revestimentos Cerâmicos
SRE	*Software Risk Evaluation*
SWEBOK	*Guide to the Software Engineering Body of Knowledge*
TBQ	*Taxonomy Based Questionary*
UML	*Unified Modeling Language*
UNESC	Universidade do Extremo Sul Catarinense
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos percebeu-se um grande crescimento na fabricação de softwares e com isso um constante avanço das técnicas computacionais. A fim de criar softwares de maior qualidade os produtores fizeram modificações nas abordagens tradicionais e, começaram a adotar controles de qualidade desde as primeiras etapas do desenvolvimento.

A Engenharia do Software propiciou o surgimento de práticas e métodos para corrigir os problemas encontrados no desenvolvimento do software. Ela encontra-se em constante desenvolvimento, devido às melhorias significantes do hardware e a evolução dos negócios. Assim os clientes tornaram-se cada vez mais exigentes e os softwares mais complexos.

A busca por padrões e ferramentas para auxiliar a equipe de desenvolvimento softwares aumentou consideravelmente nos últimos tempos, a utilização dos ciclos de vida na produção da aplicação tornou-se fundamental, para desenvolver softwares complexos de forma rápida, estruturada e com previsibilidade dos resultados obtidos. Com isso pretende-se evitar possíveis falhas e garantir um produto satisfatório de acordo com as expectativas do cliente/usuário.

Soares (2007) define que a Gerência de Riscos é o processo da Gerência de Projetos que trata dos eventos negativos que ocorrem no desenvolvimento, tais como: atrasos no prazo de entrega, elevado custo do produto, erros de requisitos fazendo com que alguns softwares, mesmo depois de finalizados, acabem não atendendo as exigências do cliente em termos de funcionalidade e qualidade.

O inicio do desenvolvimento de software trata da obtenção dos requisitos, nela deve constar basicamente às necessidades do cliente. Ela está presente em todos os modelos

de desenvolvimento de software (Cascata ou Clássico, Prototipagem, *Extreme Programming*, Espiral e outros). Porém existe uma carência das organizações em traçar planos de levantamento e gerenciamento de requisitos consistentes para servir de base para todo o desenvolvimento, pois a maneira com que os requisitos são obtidos poderá influenciar diretamente no sucesso de um projeto e determinar a qualidade do produto entregue.

Pressman (2006) define a qualidade do software como sendo a conformidade a requisitos funcionais e de desempenho declarados pelo cliente. Desta forma, torna-se necessário à elaboração de boas práticas e estudos nessa área, para gerar requisitos mais consistentes e completos. Neste sentido, Carvalho, Tavares e Castro (2004) afirmam que a indústria de software vem demonstrando um crescente interesse em Engenharia de Requisitos, isto é, entender o que se deseja construir antes de começar a construí-lo.

Desta forma, o trabalho aborda aspectos referentes ao ciclo de vida do software, mais especificamente sobre a fase de desenvolvimento de requisitos. Serão apresentados diversos processos envolvidos na produção dos requisitos e suas técnicas.

Um Projeto de Software será analisado como estudo de caso. Os processos envolvidos para a produção dos requisitos, as técnicas e os próprios requisitos serão avaliados utilizando o gerenciamento de riscos. A Gerência de Riscos além de identificar vai propor ações de combate ao mesmo controlando para que o risco seja minimizado. Assim as fases posteriores do projeto poderão contar com um documento de requisitos mais claro, consistente e completo minimizando os riscos provenientes dos requisitos.

1.1 OBJETIVO GERAL

Estudar e analisar métodos para melhorar a Engenharia de Requisitos e evitar riscos no processo de desenvolvimento do software.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) apresentar conceitos gerais da Engenharia de Software;

b) apresentar os principais modelos de Ciclo de Vida do Software;

c) compreender os principais modelos de desenvolvimento de software;

d) compreender a Engenharia de Requisitos;

e) aplicar ferramentas de Gerenciamento de Risco na Engenharia de Requisitos;

f) compreender os riscos gerados no Processo de Requisitos dos modelos de desenvolvimento de software;

g) proporcionar um estudo dos erros/riscos na fase de Engenharia de Requisitos.

1.3 JUSTIFICATIVA

A necessidade do desenvolvimento de aplicações complexas de forma rápida e eficiente para atender as exigências das organizações resultou no aumento da procura por normas e técnicas de gerenciamento de desenvolvimento do software visando melhoria contínua dos processos envolvidos.

Na busca por novas tecnologias, as empresas fabricantes de software tem que considerar a compreensão de uma série de fatores para um desenvolvimento correto. A produção dos requisitos é uma das fases que sofre com o descaso das empresas, mesmo sendo uma fase fundamental para que o Projeto de Software cresça sobre uma base sólida, que não vá gerar complicações em fases mais tardias do projeto (PRESSMAN, 1995).

As empresas acabam não utilizando padrões apropriados à estrutura do projeto ou ainda não aplicando técnicas e ferramentas para gerar uma documentação consistente, identificando o que o cliente necessita e como isso vai ser disponibilizado a ele de forma específica. Um software que não está bem especificado no documento de requisitos vai gerar problemas ao desenvolvimento que terá que adequá-lo as necessidades do cliente, posteriormente (TURINE; MASIERO, 1996).

Segundo o Standish Group (2009) pesquisas mostram que no ano de 2009 apenas 32% dos projetos foram bem sucedidos, sendo finalizados com os requisitos exigidos pelo cliente e dentro do cronograma e custo. Do restante, 44% conseguem terminar, porém não obedeceram ao cronograma, qualidade esperada e/ou custo estimado, gerando insatisfação por parte do cliente. E os 24% restantes, não conseguem nem alcançar o produto final.

Espindola, Majdenbaum e Audy (2004) afirmam que, à fase de Engenharia de Requisitos é uma das principais causas de fracassos dos Projetos de Software. Riscos não

descobertos nesta fase podem se tornar grandes problemas em fases posteriores do projeto, sendo inviável sua correção. Segundo Soares (2007) devemos tratar os riscos na origem onde o custo é menor e será mais fácil de corrigi-lo.

A fim de se obter um produto da Engenharia de Requisitos mais confiável a aplicação de processos de gerenciamento de risco torna-se uma boa solução, pois os riscos poderão ser detectados antes que se transformem em problemas. Segundo Pressman (1995) o gerenciamento de riscos tem o objetivo de melhorar a qualidade do produto e do processo de desenvolvimento de software.

Sendo assim, pretende-se estudar e compreender conceitos de gerenciamento de riscos e aplicá-los a um Projeto de Software para detectar e corrigir os riscos produzidos nos requisitos. Com isso será proporcionado um escopo de riscos de requisitos visto na prática e apresentar soluções que contribuam para a formação de um conjunto de requisitos consistente, claro e confiável, aumentado às chances de êxito do produto.

1.4 ESTRUTURA DO TRABALHO

O trabalho é constituído de oito capítulos, sendo o primeiro composto por uma introdução a pesquisa desenvolvida, seguido da descrição dos objetivos e a justificativa proposta para a realização do trabalho desenvolvido.

O Capítulo 2 apresenta uma breve introdução sobre a Engenharia de Software e parte do embasamento teórico referente ao processo de desenvolvimento de software ou ciclo de vida do software. Questões relacionadas à qualidade do software e algumas estatísticas de riscos em Projetos de Software também foram abordados neste capítulo.

No Capítulo 3 são abordados os tipos de requisitos e as fases da Engenharia de Requisitos, esse capítulo serve de base para o desenvolvimento do estudo de caso. Neste

capítulo também é feita a descrição de como deve ser um documento de requisitos, esse documento reuni todos os requisitos identificados na Engenharia de Requisitos.

O Capítulo 4 vai apresentar os principais modelos de ciclo de vida do software e explicar como eles funcionam. A seguir, o capítulo 5 abordará os conceitos, normas e fases da Gerência de Riscos. Este capítulo vai evidenciar as atividades utilizadas no desenvolvimento do estudo de caso.

O Capítulo 6 relaciona alguns trabalhos utilizados para compreensão e conhecimento na área de Engenharia de Requisitos, Qualidade de Software e Gerência de Riscos, além de técnicas, modelos e ferramentas utilizadas para o gerenciamento dos requisitos.

O trabalho desenvolvido é apresentado no Capítulo 7, sendo relatada as etapas de metodologia utilizadas no desenvolvimento da pesquisa proposta, associados aos resultados obtidos. Por fim, encontra-se a conclusão deste trabalho e a sugestão de temas referentes a pesquisas futuras na área de qualidade de produção de software.

2 ENGENHARIA DE SOFTWARE

A Engenharia de Software (ES) é a área da computação que busca corrigir os problemas no desenvolvimento do software. Ela utiliza modelos, métricas e ferramentas que auxiliam na otimização do processo de desenvolvimento. Nesse contexto, o desenvolvimento deve seguir uma estratégia, com atividades que devem garantir o sucesso do produto final (PRESSMAN, 1995).

O objetivo proposto pela ES é a de melhorar a qualidade dos softwares, sobretudo dos complexos, e com isso aumentar a produtividade, confiabilidade e diminuir custos (REZENDE, 2005).

2.1 HISTÓRICO

O software de computador é, atualmente, uma tecnologia importante para o mundo, tendo um crescimento rápido desde 1950. Com modernização dos negócios e o desenvolvimento do hardware de terceira geração, houve a necessidade de se desenvolver aplicações maiores e mais complexas e, assim começou a surgirem problemas relacionados a correções, adaptações, atrasos no prazo de entrega, difícil manutenção e baixo desempenho, essa fase ficou conhecida como a “crise do software”, em 1970 (SOMMERVILLE, 2003).

Foi preciso criar novas técnicas e métodos para que essa nova complexidade, proporcionada pelos softwares, fossem tratadas. Com isso os gerenciadores abordaram o desenvolvimento de software com princípios e paradigmas de outras engenharias, existentes na época. Assim surgiu a ES, uma aplicação concebida por meio de um processo sistemático,

disciplinado e com padrões de qualidade à concepção, implementação e manutenção (IEEE, 1998).

2.2 PROCESSO DE DESENVOLVIMETO DE SOFTWARE

O Processo de Desenvolvimento de Software (PDS) é definido por Pressman (2006) como sendo um conjunto de tarefas que são necessárias para construir um software de qualidade e reduzir custos. Os processos de desenvolvimento têm como objetivo o aperfeiçoamento do software que será produzido, gerando para o cliente produtos personalizados que buscam satisfazer suas necessidades.

Segundo Sommerville (2003) o PDS está dividido em sub-processos ou atividades que buscam detalhar o que será feito, que passos deverão ser percorridos, por quais agentes, que tecnologias serão utilizadas até a elaboração do produto final, são eles:

a) *especificação de software*: equivale a fase de obtenção dos requisitos;
b) *projeto e implementação de software*: desenvolvimento do software propriamente dito;
c) *validação de software*: verificação do cliente se o software atende as necessidades;
d) *evolução do software*: atender a novos requisitos descobertos.

Os processos tornam o desenvolvimento gerenciável, proporcionando estabilidade e organização a está fase. Os modelos de desenvolvimento de software representam cada um, um conceito particular para o desenvolvimento, portanto os gestores precisam ter experiência

e criatividade para adaptar os processos de software aos softwares a serem desenvolvidos, e contribuir para o sucesso do produto final (SOMMERVILLE, 2003).

Existem vários tipos de modelos de desenvolvimento, e os principais modelos serão abordados no Capítulo 5. A utilização desses modelos comprovados cientificamente tem a finalidade de melhorar, dar mais qualidade ao desenvolvimento e conseqüentemente produzir um software que atenderá as necessidades do cliente com qualidade.

2.3 QUALIDADE DE SOFTWARE

Pressman (2006) define a qualidade de um software como concordância entre os requisitos funcionais e de desempenho estabelecidos pelo cliente. A fase de requisitos pode ser considerada uma atividade, do PDS, primordial para que se tenha sucesso no produto final. Segundo Sayão, Staa e Leite (2003) a preocupação com a qualidade na produção de software inicialmente centrou suas atenções na qualidade do produto final, mas está visão evoluiu e atualmente a preocupação com qualidade envolve também as fases que compõe o desenvolvimento do software.

A qualidade do software pode ser controlada através de atividades como inspeções, revisões e testes realizados ao longo do PDS para garantir que os requisitos do cliente e de software estejam em conformidade. Quando não se tem a conformidade, deve-se iniciar um ciclo de realimentação no PDS, garantindo que os requisitos sejam atendidos. A qualidade pode ser mesurada a partir da eficiência, erros encontrados/horas trabalhadas, e da confiabilidade que é a probabilidade de o software não apresentar falhas em certo período de tempo essas métricas são utilizadas quando se tem uma base de dados com conhecimentos passados de outros desenvolvimentos (SAYÃO; STAA; LEITE, 2003).

Fabricar um software de qualidade é importante, mas a satisfação do cliente com o software lhe oferecendo benefícios consideráveis tem maior importância, pois diante de um software que atende as suas necessidades os clientes são capazes de tolerar problemas de confiabilidade ou desempenho (PRESSMAN, 2006).

2.4 ESTATÍSTICA DE RISCOS DE SOFTWARE

Pesquisas mostram que no ano de 2009 apenas 32% dos projetos foram bem sucedidos, sendo finalizados com os requisitos exigidos pelo cliente e dentro do cronograma e custo. Do restante, 44% chegaram ao fim, porém não obedecendo ao cronograma, qualidade esperada e/ou custo estimado, gerando insatisfação por parte do cliente. E os 24% restantes, não conseguem nem alcançar o produto final, conforme Figura 1(STANDISH GROUP, 2009).

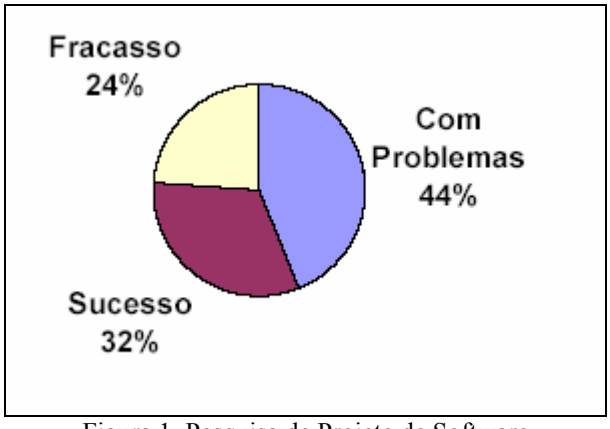


Figura 1. Pesquisa de Projeto de Software
Fonte: Adaptado de STANDISH GROUP (2009)

Estudos comprovam que dos Projetos de Software (PS) concluídos, apenas uma pequena parte corresponde às expectativas, tendo-se observado que o problema se centra principalmente numa deficiente análise de requisitos (GRUNBACHER; BRIGGS, 2001).

Mesmo sendo um fator essencial para se fabricar softwares corretos e satisfatórios, o processo de requisitos ainda fica em segundo plano em muitas empresas. Os fatores que contribuem para que o processo de requisitos seja falho são: às pressões de mercado, à alta competitividade entre empresas e o tamanho das empresas, que não dispõem de recursos humanos em quantidade e qualidade suficientes.

A partir de um estudo realizado por Jones (1996) descobriu-se que o processo de requisitos esteve deficiente em mais de 75% das organizações. Segundo Zahedi (1995) os erros na fase de produção de requisitos consomem 82% dos recursos gastos em correção de erros nos projetos, enquanto que erros no design consomem 13%, erros na codificação consomem 1%, e nas demais fases consomem 4%.

3 ENGENHARIA DE REQUISITOS

O software é um produto que tem um objetivo e uma necessidade de criação, não se cria um software que não tem utilidade. Assim, um software deve ser criado para atender as expectativas de um cliente em potencial (PRESSMAN, 2006). As atividades que determinam os objetivos de um software e as restrições associadas a ele são conhecidas como processo de requisitos, os sub-processos de descobrir, analisar, documentar e verificar essas características é conhecido como Engenharia de Requisitos (ER) (SOMMERVILLE, 2003). Ela deve também, estabelecer o relacionamento entre estes objetivos e restrições e a especificação precisa do software.

A importância e complexidade deste processo levaram ao surgimento, no início dos anos 90, da ER que é o processo de aquisição, refinamento e verificação das necessidades do usuário (IEEE, 1998). A ER tem o objetivo de sistematizar o processo de definição dos requisitos, obtendo uma especificação correta e completa, tornando mais eficaz o documento de requisitos e mais eficiente o processo utilizado para produzi-lo. Boehm (1989) define a ER como uma disciplina cujo objetivo é desenvolver uma especificação completa, consistente e não-ambígua, servindo de base para um acordo entre todas as partes envolvidas e descrevendo as necessidades que o produto de software resolverá ou executará. Ela requer fundamentação e processos próprios, e que devem ser planejados e gerenciados ao longo de todo o ciclo de vida (SOMMERVILLE, 2001).

O processo de ER descreve um conjunto estruturado de atividades a serem seguidas para criar, validar e manter um documento de requisitos. Algumas organizações têm seu processo de ER explicitamente definido e padronizado. Sommerville e Sawyer (1997)

sugerem que cada organização deve desenvolver um processo adequado à sua realidade. Kotonya, Sommerville (1998) definem um modelo genérico de atividades que podemos ver na Figura 2:

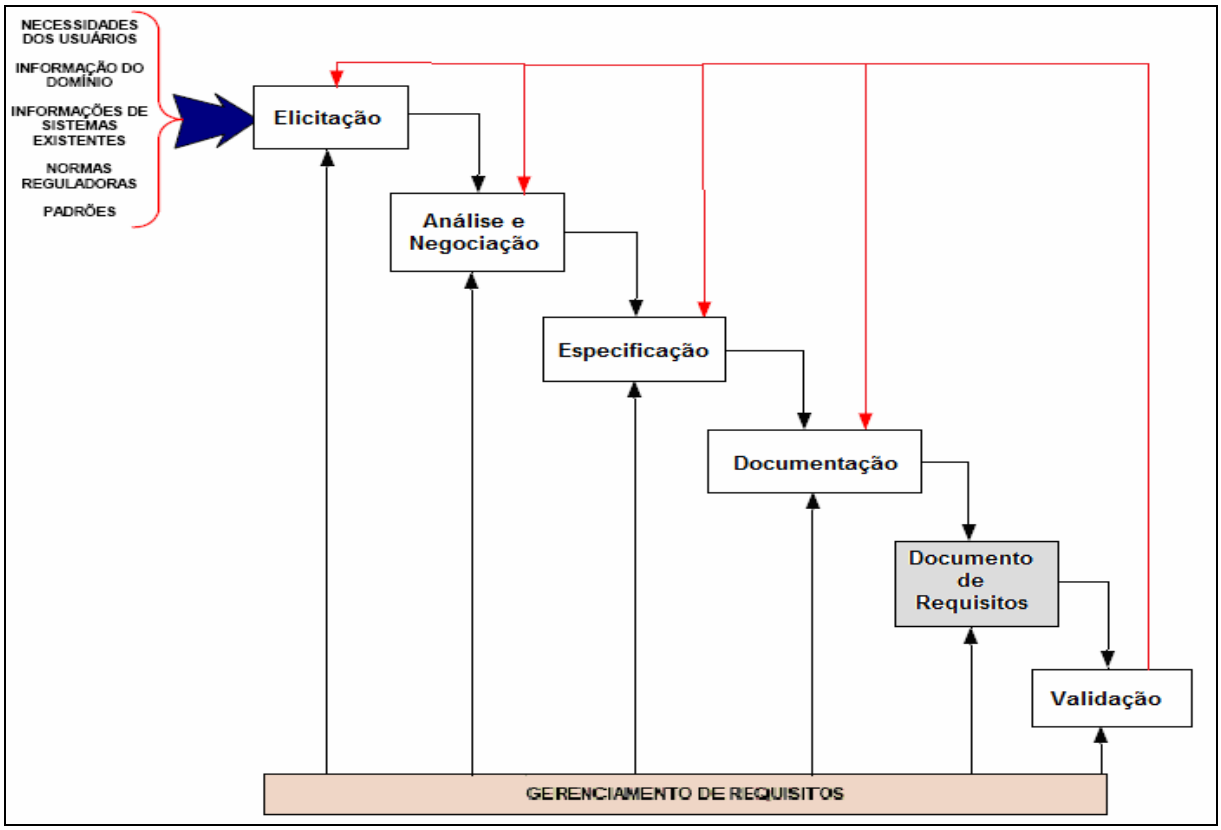


Figura 2. Um modelo para o processo de Engenharia de Requisitos
Fonte: Adaptado de LOPES, P.S.N.D, (2002)

Podemos definir como entrada as informações gerais sobre o domino do software, normas e regulamentos externos, padrões a serem seguidos pelo desenvolvimento e recursos necessários para se produzir o software.

O termo elicitação não existe na língua portuguesa, mas foi criada e tem sido utilizada por engenheiros de requisitos. Significa obter e tornar explícito o máximo de informações possíveis para o conhecimento de um objeto em questão. Na etapa de elicitação dos requisitos os engenheiros de requisitos devem levantar as expectativas e necessidades dos

clientes/usuários com relação ao software a ser desenvolvido, os requisitos obtidos nesta fase vão servir de base para a análise (ESPINDOLA; MAJDENBAUM; AUDY, 2004).

A análise distribui os requisitos em categorias, explora as relações entre eles, e identifica o grau de importância de cada um, de acordo com as necessidades dos clientes/usuários. Por sua vez, os requisitos são negociados para decidir quais devem ser aceitos, de forma a obter consenso, na documentação ocorre à especificação dos requisitos. Por final, acontece a validação dos requisitos que examina a especificação dos requisitos para garantir que todos os requisitos foram definidos de forma clara, consistente e sem ambigüidades ou omissões, e que todos os erros foram corrigidos. Na prática existe muita sobreposição e interação entre uma atividade e outra (ESPINDOLA; MAJDENBAUM; AUDY, 2004).

Segundo Espindola, Majdenbaum e Audy (2004) os benefícios da ER são: concordância entre desenvolvedores, clientes e usuário sobre o trabalho a ser feito e quais os critérios de aceitação do sistema, uma base precisa para a estimativa dos recursos, melhoria nas qualidades do sistema e alcançar os objetivos com o mínimo de desperdício.

O principal artefato realizado na ER é o documento de requisitos, e de acordo com Sommerville e Sawyer (1997), o documento de requisitos é uma declaração formal dos requisitos para os *stakeholders*, que podem ser clientes, usuários finais ou a equipe de desenvolvimento do software. O documento de requisitos pode receber vários nomes, como “especificação funcional”, “definição de requisitos” ou ainda “especificação dos requisitos do software”, e em algumas organizações como o Departamento de Defesa dos EUA e o *Institute of Electrical and Electronics Engineers* (IEEE) elas definem seus próprios padrões para o documento de requisitos (ESPINDOLA; MAJDENBAUM; AUDY, 2004).

Como visto, a maior parte dos problemas no desenvolvimento de software é originada justamente na etapa de levantamento e definição dos requisitos. Quando um erro de requisitos é descoberto em fases mais adiantadas do PS o custo da correção torna-se mais alto e mais difícil de ser realizado, pois diversas fases certamente precisaram de mudanças para atender os novos requisitos. Segundo Blackburn, Busser e Nauman (2001), corrigir um erro de requisitos quando o software estiver em testes pode ser até 100 vezes mais caro que descobrir e corrigir o problema na fase de requisitos.

Com o objetivo de diminuir os riscos dos requisitos e obter um *feedback* mais cedo durante o processo de desenvolvimento, muitas organizações resolveram adotar um processo de desenvolvimento iterativo e evolucionário, como os modelos de ciclo de vida Espiral e o *Rational Unified Process* (RUP). Posteriormente, surgiram as tecnologias ágeis que visam à interação com o cliente ao invés de documentação, mas não geram uma previsibilidade eficiente.

3.1 REQUISITOS

Os requisitos de software devem expressar as características do produto (funções, recursos a serem disponibilizados e limitações), sendo que, os requisitos têm o objetivo de garantir que as necessidades do cliente sejam satisfeitas. Os requisitos são a base para a definição da arquitetura do software, para a implementação, geração dos casos de testes e para a validação do sistema junto ao cliente (SAYÃO; LEITE, 2006).

Entender os requisitos de um problema é uma das tarefas mais difíceis enfrentadas por um engenheiro de software. São varias as formas de obtenção dos requisitos como: leitura de manuais ou de reuniões com o cliente, entrevistas, questionários onde eles vão expressar

suas necessidades, as visitas a ambiente e instalações para acompanhar suas rotinas podem ser utilizada para ajudar a fornecer um entendimento do trabalho. O engenheiro de requisitos necessita compreender os requisitos, mesmo não tendo experiência na área de domínio. Quando isso não acontece existe o risco de haver distúrbios na comunicação fazendo com que os requisitos sejam compreendidos incorretamente (PRESSMAN, 2006).

Compreender o domínio do problema é necessário para implementar as rotinas e procedimentos que o software realizará. As falhas ocorridas no levantamento e especificação dos requisitos, não detectadas, podem gerar um software que não irá atender todas as necessidades do cliente. Os problemas ocasionados pela fase de requisitos são uma das principais causas de fracassos de projetos de software.

A fase de requisitos pode ser vista em três níveis de descrição (SOMMERVILLE, 2003):

a) **requisitos de usuário:** linguagem do cliente e uso de diagramas, para exemplificar, funções e restrições. Alto nível de abstração;
b) **requisitos de sistema:** detalhamento das funções e restrições do sistema. O documento produzido, neste nível, deve ser preciso, pois poderá servir de contrato entre contratante e desenvolvedor;
c) **especificação de projeto:** acrescenta-se detalhes mais específicos aos requisitos do sistema. Serve de base para projeto e implementação.

Os requisitos devem formar uma série de sentenças que descrevem de maneira clara, concisa, consistente e não-ambígua todas as informações referentes ao sistema (ROCHA; MALDONADO; WEBER, 2001). Segundo Gustafson (2003) os requisitos são

aspectos essenciais necessários para o software ser produzido. A seguir vamos apresentar os três tipos de requisitos principais.

3.1.1 Requisitos Funcionais

Os Requisitos Funcionais declaram quais serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. De um modo geral os requisitos funcionais definem os recursos específicos que devem ser fornecidos pelo software (SOMMERVILLE, 2003).

Existe uma grande quantidade de problemas da ES tem origem na especificação de requisitos, por má interpretação do desenvolvedor que pensa em simplificar o desenvolvimento. Os requisitos funcionais devem ser especificados de forma consistente e completa no documento de requisitos. Mas na prática esses aspectos se tornam quase impossíveis de serem alcançados se tratando de desenvolvimento de softwares grandes e complexos (SOMMERVILLE, 2003).

A falta de análises mais detalhadas dos requisitos poderá gerar discordâncias entre a gerência por se tratar de informações inconsistentes. A implementação de uma gerência de riscos nesta fase ajudará melhorando a qualidade das informações e consequentemente a do software. São exemplos de requisitos funcionais:

a) o software deve possibilitar o cálculo dos gastos diários, semanais, mensais e anuais com pessoal;
b) o software deve emitir relatórios de compras a cada semana;
c) os usuários devem poder obter o número de aprovações, reprovações e trancamentos em todas as disciplinas por um determinado período de tempo.

3.1.2 Requisitos Não-Funcionais

Os Requisitos Não-Funcionais identificam as condições de comportamento do software que não são funcionais. Esses requisitos estão relacionados à qualidade, restrições operacionais ou do desenvolvimento do software, e muitos descrevem características do software como um todo, então, a falha de um requisito poderá ocasionar o fracasso do software (SOMMERVILLE, 2003).

O surgimento dos requisitos não-funcionais ocorre de acordo com as necessidades do cliente em relação ao software. Os impactos causados por restrições operacionais ao software ou por específicos comportamentos, mostraram-se um grande desafio a ser contornado para que se tenha softwares que satisfaçam seus requisitos não funcionais e sejam desenvolvidos em tempo hábil (CHICHINELLI; CAZARINI, 2001). A Figura 3 demonstra a classificação de diferentes tipos de requisitos não funcionais.

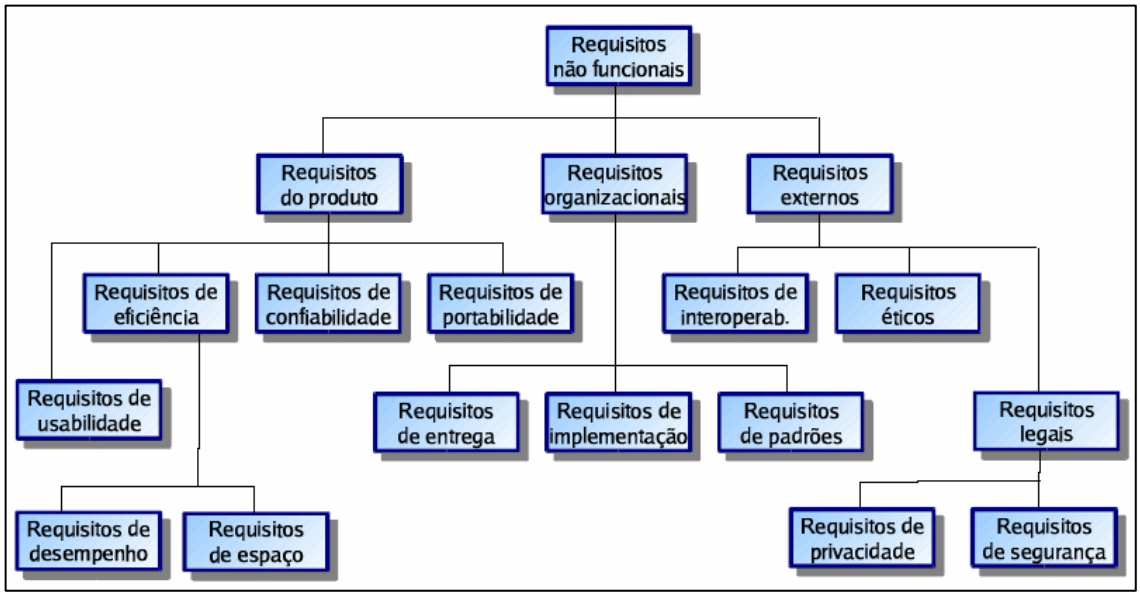


Figura 3. Requisitos Não-Funcionais
Fonte: SOMMERVILLE, I. (2003)

Sommerville (2003) classifica os diferentes tipos de requisitos não funcionais em três áreas:

a) *requisitos de produto:* são aqueles que vão especificar o comportamento do produto;
b) *requisitos organizacionais:* dizem respeito a procedimentos de política e organizacional do cliente e desenvolvedor;
c) *requisitos externos:* são os requisitos procedentes de fatores externos ao desenvolvimento e ao software.

Normalmente estes requisitos são descritos de maneira informal ou de maneira controversa (por exemplo, o cliente quer segurança, mas os usuários querem usabilidade) e são difíceis de validar. A identificação nem sempre é uma tarefa fácil, pois eles podem descrever os objetivos gerais do cliente, e por causa dessa amplitude causar problemas de interpretações e discussões (SOMMERVILLE, 2003). Um cheklist pode ser usado como inicio ao levantamento desses requisitos não-funcionais, conforme a Figura 4. Assim o cliente poderá entender o que claramente o projetista necessita identificar e descrever quais as características que ele espera que o software tenha, e ajudando a não deixar o escopo ficar vago. Exemplos podem ser adicionados para facilitar o entendimento.

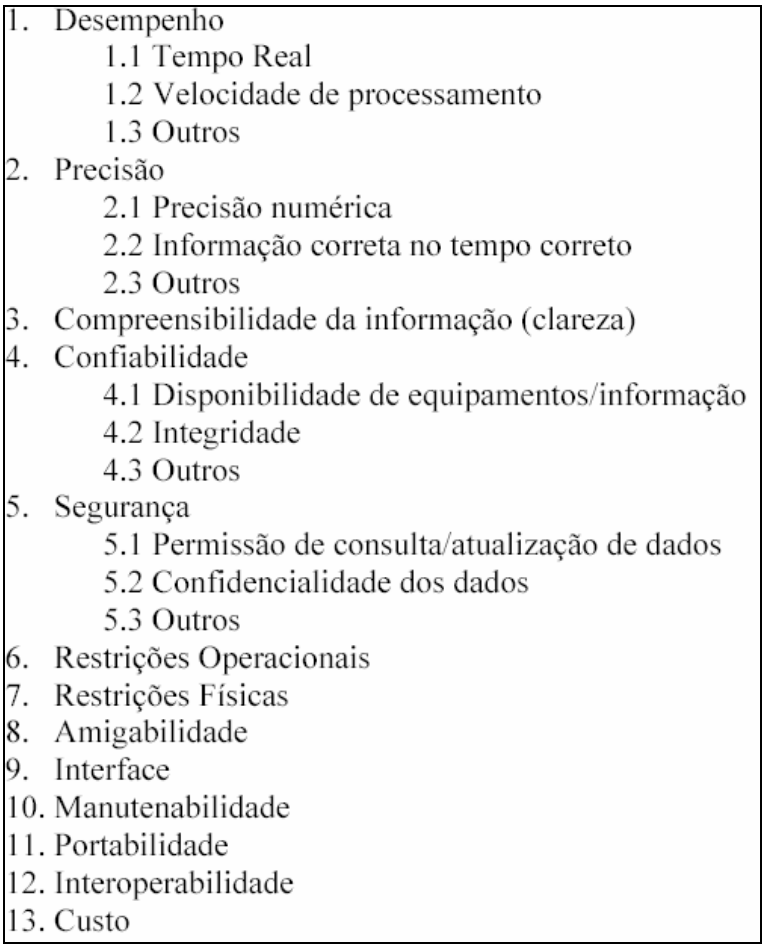


Figura 4. Checklist de Requisitos Não-Funcionais
Fonte: CYSNEIROS, L.M.; LEITE, J.C.S.P. (1997)

Abaixo pode ser observado alguns exemplos de requisitos não-funcionais:

a) a base de dados deve ser protegida para acesso apenas de pessoas autorizados;

b) o tempo de resposta do sistema não deve ultrapassar 30 segundo;

c) o software deve ser operacional no sistema *WindowsSever* 2ooo;

Sommerville (2003) identifica que existem vários casos onde fica evidente que a falta de um tratamento adequado dos requisitos não funcionais leva a resultados desastrosos, tratar os riscos dessa área é essencial para o sucesso do software.

3.1.3 Requisitos de Domínio

Segundo Sommerville (2003) os requisitos de domínio nascem a partir do domínio da aplicação do sistema e refletem características dele, ao contrário dos requisitos funcionais e não-funcionais eles não derivam das necessidades dos clientes, porém eles podem ainda se transformar em requisitos funcionais ou limitar os já existentes.

Os requisitos referentes ao domínio são muito sensíveis a interpretações, pois são escritos por especialistas do ambiente da aplicação em uma linguagem natural dificultando a análise do desenvolvedor, que pode desconhecer completamente o domínio do problema (SOMMERVILLE, 2003). Abaixo se observa exemplos de requisitos do domínio:

a) a soma dos percentuais a ser distribuído entre os fundos incluídos no plano de aplicação deve ser entre 0 e 100%;
b) os campos referentes à “orçamento de projeto vinculado” só estarão ativos se o tipo de projeto for vinculado.

Pode ocorrer vícios na especificação dos requisitos de domínio por parte dos especialistas que não tornam todas as informações explicitas por acharem que algumas são obvias de mais, levando ao desenvolvedor a implementar requisitos incorretos (SOMMERVILLE, 2003). A necessidade de uma análise de riscos na produção desses requisitos é evidente, pois eles identificam aspectos relacionados ao negócio e se não forem desenvolvidos corretamente podem gerar um software de baixa utilidade.

3.2 CONCEPÇÃO E ESTUDO DE VIABILIDADE

A produção de um software precisa ter um objetivo ou necessidade. A maioria dos softwares teve seu inicio a partir de uma necessidade de negócio ou um mercado, ou serviço novo que tenha um alto grau de desenvolvimento. Assim com uma idéia defini-se um caso de negócio e procura-se identificar características do domínio e assim estabelecer uma compreensão em alto nível do problema como quem vai utilizar, a natureza da solução e a efetividade da comunicação (PRESSMAN, 2006).

Segundo Sommerville (2003) deve-se criar um estudo de viabilidade que vai coletar e avaliar as informações, utilizando questionários, e produzir um relatório identificando se o desenvolvimento de um software para solucionar tal problema deve ser produzido ou não. O estudo de viabilidade deve abordar três aspectos principais:

a) *viabilidade econômica*: quanto a organização está disposta a investir para a obtenção do software;
b) *viabilidade técnica*: se a linguagem, o banco e as técnicas para codificação são possíveis de serem utilizadas e implementadas;
c) *viabilidade legal*: se a construção do software, bem como suas funções, não infringem leis ou normas da organização ou até mesmo da constituição.

3.3 LEVANTAMENTO DE REQUISITOS

O levantamento ou elicitação de requisitos como também é conhecida evolve a captação dos requisitos do software, buscando obter conhecimento do domínio do problema e como o software será usado diariamente, desempenho exigido e as restrições do hardware. As atividades principais do levantamento de requisitos consistem na identificação da fonte, coleta da informação (SOMMERVILLE, 2003). O trabalho de levantamento de requisitos é realizado em conjunto com os clientes usuários e colaboradores do software. Em reuniões ministradas e assistidas por engenheiros de software ou clientes procura-se identificar o problema, propor soluções, negociar diferentes abordagens e especificar um conjunto inicial de requisitos (PRESSMAN, 2006).

O levantamento de requisitos deve envolver diversas pessoas, o termo *stakeholder* é utilizado para qualquer pessoa, considerada relevante, que tiver influência sobre os requisitos. A identificação das fontes, *stakeholder*, é necessária para a compreensão do domínio do problema, pois cada um se expressa de acordo com seu ponto de vista (SOMMERVILLE, 2003).

A coleta das informações normalmente é feita através de reuniões e entrevistas com os *stakeholders*, além de questionários. Mas observa-se que a utilização de reuniões e entrevistas nem sempre são suficientes para se obter os requisitos necessários para o desenvolvimento do software. Assim o desenvolvedor deve aplicar outras técnicas de coleta, em paralelo com as reuniões e entrevistas, para obter um conjunto de requisitos satisfatório. Existem outras técnicas para coletar e analisar requisitos como: o levantamento orientado a pontos de vista, o uso de cenários e a etnografia, além dos métodos de análise estruturada e a prototipação (SOMMERVILLE, 2003).

A participação dos clientes e usuários tem que ser ativa, para que os engenheiros de requisitos se comuniquem com eles e compreendam o domínio do problema. Sommerville (2003) define um processo genérico de levantamento e análise de requisitos em seis etapas:

a) **compreensão do domínio:** entender como o software irá funcionar;
b) **coleta de requisitos:** interação com os *stakeholders* para descobrir suas necessidades;
c) **classificação:** organizar os requisitos em grupos de semelhanças;
d) **resolução de conflitos:** procura solucionar os requisitos que estão em conflito;
e) **definição das prioridades:** interação com os *stakeholders* para definir os mais importantes;
f) **verificação de requisitos:** garantir que os requisitos expressem as necessidades dos *stakeholders* de forma clara e consistente.

O levantamento de requisitos, aparentemente, parece ser um processo simples. Porém na prática nota-se que isso não é verdade, pois os clientes podem não saber o que querem realmente, se expressando de forma geral ou acabam omitindo informações por acharem que são óbvias de mais dificultando a compreensão do engenheiro (PRESSMAN, 2006).

Como existem usuários diferentes, podem existir necessidades diferentes, e cabe ao engenheiro de requisitos descobrir esses diferentes usuários e achar os pontos comuns e conflitantes. Fatores políticos, econômicos e de negócio também influenciam o levantamento de requisitos, por isso devem ser considerados com atenção (SOMMERVILLE, 2003).

Os riscos estão presentes no levantamento de requisitos e devem ser acompanhados de perto pelos engenheiros para que problemas no escopo com o limite mal

definido, por parte do usuário, com especificações técnicas desnecessárias, confundindo o engenheiro, falta de entendimento dos clientes/usuários por terem pouca compreensão do domínio – especificam requisitos conflitantes, ambíguos ou impossíveis de testar – dificuldades em informar as suas necessidades e as mudança dos requisitos não proporcionem o insucesso do software (PRESSMAN, 2006).

3.3.1 Características de Qualidade dos Requisitos

O fato dos requisitos sofrerem mudanças e que novos requisitos podem surgir ao longo do desenvolvimento torna o esforço ainda maior para que o produto final do processo de requisitos forneça a maior quantidade possível de informações sobre o software. Contudo, as informações devem ser confiáveis, e para que isso aconteça, elas têm que seguir características de qualidade. Os requisitos de alta qualidade são claros, completos, sem ambigüidade, programáveis, consistentes e testáveis. Abaixo podemos ver outros aspectos que os requisitos de qualidade devem possuir (PAULA FILHO, 2001):

a) **correta:** o requisito expresso é um requisito do software a ser construído;
b) **precisa:** possui apenas uma interpretação;
c) **completa:** todas as decisões de especificação foram tomadas;
d) **consistente:** não existem conflitos com outros requisitos;
e) **priorizada:** classificar o requisito de acordo com a sua importância, estabilidade e complexidade;
f) **verificável:** os requisitos são testáveis;
g) **modificável:** a estrutura permite a mudança do requisito facilmente e de forma consistente;

h) **rastreável:** fácil determinação dos antecedentes e das conseqüências dos requisitos.

Os requisitos que não possuírem esses aspectos de qualidade correm um risco maior de apresentarem problemas e devem ser revistos ou negociados.

3.3.2 Levantamento de Requisitos Orientado a Pontos de Vista

O levantamento orientado a pontos de vista busca uma forma de levantar os requisitos representando as perspectivas de diferentes *stakeholders* relevantes ao software, que depois podem ser classificados considerando as diferenças. A ER utiliza está técnica não apenas para levantar requisitos, mas também para estruturar e organizar o processo de levantamento (SOMMERVILE, 2003).

Os pontos de vista geram análise de visão múltipla que se torna importante visto que não existe uma única forma correta de analisar os requisitos do software (PRESSMAN, 2006).

3.3.3 Cenários

Os Cenários descrevem uma narrativa ou um pequeno conjunto de narrativas da situação do domínio na prática que favorecem o esclarecimento das informações, a identificação de problemas e a antecipação das soluções (CALDAS JUNIOR; MASIERO, 1998). O uso de cenários é uma maneira excelente de representar, para clientes e usuários, os

problemas atuais, as possibilidades e detalhes que podem surgir na especificação de requisitos (SOMMERVILLE, 2003).

A utilização dos Cenários tem como foco representar as atividades que as pessoas realizam nas organizações, proporcionando exemplos da vida real e possibilitando uma perspectiva mais ampla dos problemas atuais onde o sistema será inserido.

Os Cenários de Eventos devem ser usados para documentar o comportamento do software quando submetido a eventos particulares, geralmente, são aplicados à descrição do fluxo de dados e as ações do sistema (SOMMERVILLE, 2003). As exceções que surgirem também pode ser documentado com o uso dos Cenários. Existem vários tipos de caminho a percorrer para formar um cenário, o caminho sem problemas ou Cenário Básico, como também é conhecido, é um deles. Nogueira (2006) identifica um exemplo, a abertura de uma conta bancária, os passos a seguir são:

a) o cliente solicita a abertura da conta;
b) o gerente aprova a abertura;
c) o gerente abre a conta;
d) o cliente realiza o primeiro depósito;
e) o cliente ganha o seu cartão magnético.

Se no Cenário Básico acima, o gerente encontrasse o nome do cliente em algum órgão de proteção ao crédito e não abrisse a conta. Este outro cenário formado seria reconhecido como Cenário Alternativo, onde ocorre algum problema no fluxo do Cenário Básico (NOGUEIRA, 2006). Outra forma de visualizar os requisitos é utilizando os diagramas da *Unfied Modeling Linguage* (UML). A notação UML oferece diversos de diagramas que complementam o entendimento dos requisitos levantados textualmente.

3.3.4 Diagramas UML

Os diagramas da UML podem ser utilizados para adicionar detalhes às funções do software, mostrando a seqüência de processamento de eventos no software (SOMMERVILLE, 2003). A UML permite ao usuário e ao analista a visualização sob diferentes perspectivas melhorando a compreensão e participação dos envolvidos no processo. Deve-se utilizar quantos diagramas for necessários para compreensão do software e seu desenvolvimento. A quantidade e quais diagramas serão utilizados deve ser definida pelo projetista e pela complexidade do software. Existem nove tipos diferentes de diagramas UML, são eles:

a) diagrama de caso de uso;
b) diagrama de classes;
c) diagrama de objeto;
d) diagrama de estado;
e) diagrama de seqüência;
f) diagrama de colaboração;
g) diagrama de atividade;
h) diagrama de componente;
i) diagrama de implantação ou execução.

O diagrama de caso de uso, Figura 5, é usado para modelar como um software ou empresa funciona, ou como os usuários desejam que ele funcione. Os casos de uso são à base da orientação a objetos na UML. No diagrama de caso de uso temos os atores externos os casos de uso e o relacionamento. Os atores representam o papel de uma entidade externa ao

sistema como um usuário, um hardware, ou outro sistema que interage com o software. Os atores se relacionam com os casos de uso, que podem ser processos, funções automáticas ou manuais. O relacionamento pode acontecer entre os próprios casos de uso também (ALMEIDA; DAROLT, 2001).

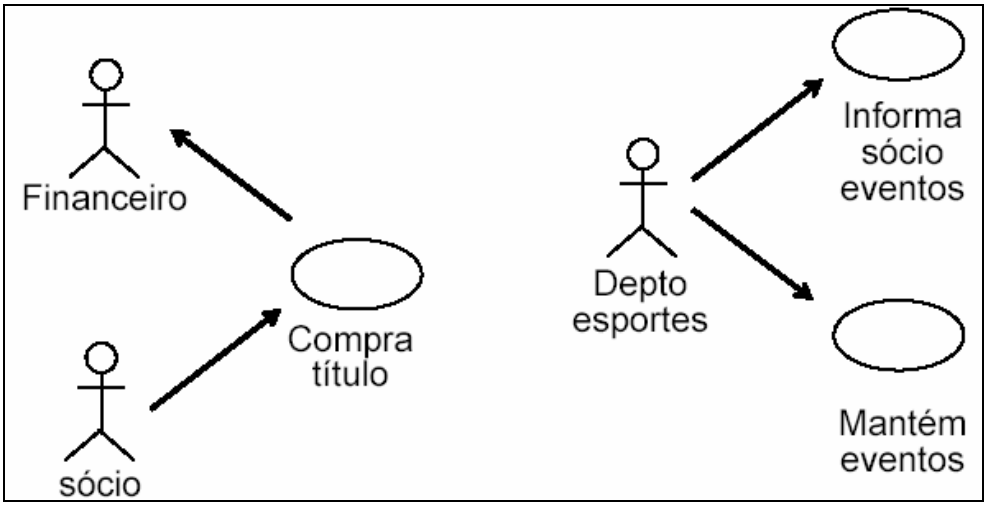


Figura 5. Casos de Uso
Fonte: CALDAS, P. O. (2009)

A descrição do caso de uso, Figura 6, tornar-se primordial para a compreensão do diagrama sendo que ela possui outras informações complementares que ajudam a compor o software.

ordem	Nome
1	Acesso ao sistema
Quem inicia	Ator funcionário setor esportes
Pré-condição	Deve estar cadastrado
Cenário 1: Acesso ao sistema	1 – funcionário informa o nome
	2 – se o nome não for cadastrado, o sistema permite informar 3 nomes inválidos, após encerra.
	3 –funcionário informa a senha
	4 – se a senha não for cadastrada, o sistema permite informar 3 senhas inválidas, após encerra
	5 – o sistema abre o menu do funcionário
Exceção	Informar usuário ou senha inválidos

Figura 6. Descrição do Caso de Uso
Fonte: CALDAS, P. O. (2009)

Os diagramas de classes, Figura 7, mostram as diferentes classes que formam um software e como elas se relacionam. Eles são chamados diagramas estáticos porque mostram as classes, com seus métodos e atributos bem como os relacionamentos estáticos entre elas, mas não mostram a troca de mensagens. Uma classe em UML é representada por uma caixa retangular com três compartimentos: um com o nome da classe, o outro com a lista de atributos da classe e o último com a lista de operações da classe ou métodos.

As associações representam relacionamentos estruturados entre os objetos de diferentes classes que podem ser: normal, recursiva, qualificada, exclusiva, ordenada, de classe, ternária, agregação e generalização (ALMEIDA; DAROLT, 2001).

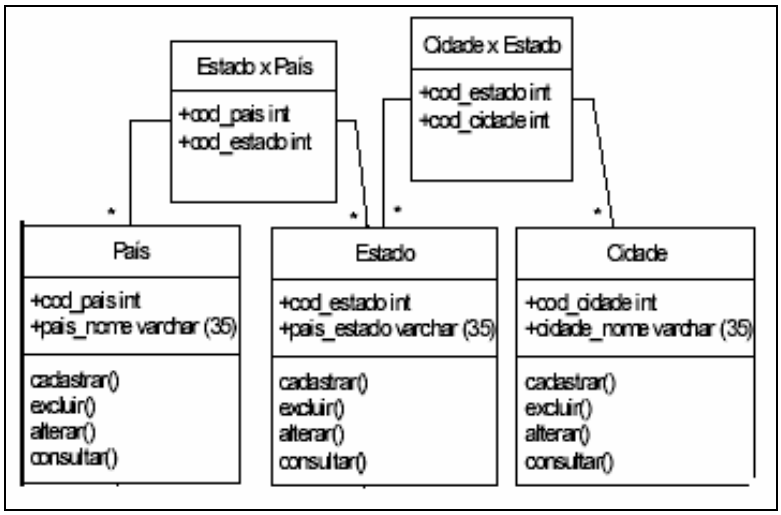


Figura 7. Diagrama de Classe

Os diagramas de objetos, Figura 8, instanciam os itens presentes no diagrama de classe. Ele é um perfil do software em um dado momento e é usado para modelar a visão de processo estática a partir de instâncias reais ou de protótipos (ALMEIDA; DAROLT, 2001).

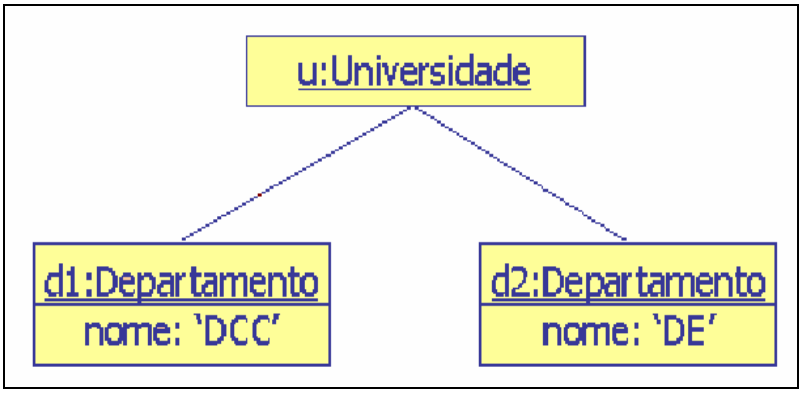


Figura 8. Diagrama de Objetos
Fonte: ALMEIDA, A.; DAROLT, R. (2001)

Os diagramas de estados são utilizados para modelar o comportamento dinâmico do software, eles capturam o ciclo de vida dos objetos mostrando os estados que um objeto pode possuir e como os eventos esses estados. Estes diagramas não são escritos para todas as classes do software, pois é preciso ter estados definidos e conhecidos, além do comportamento ser afetado e modificado (RUMBAUGH; JACOBSON, 2000).

O diagrama de estado, Figura 9, possui um ponto de início, mas pode ter vários pontos de finalização. Os estados representam as condições em que se encontram os objetos. Os eventos são os causadores de mudança de um estado para outro. As linhas de transição representam a troca de um estado (ALMEIDA; DAROLT, 2001).

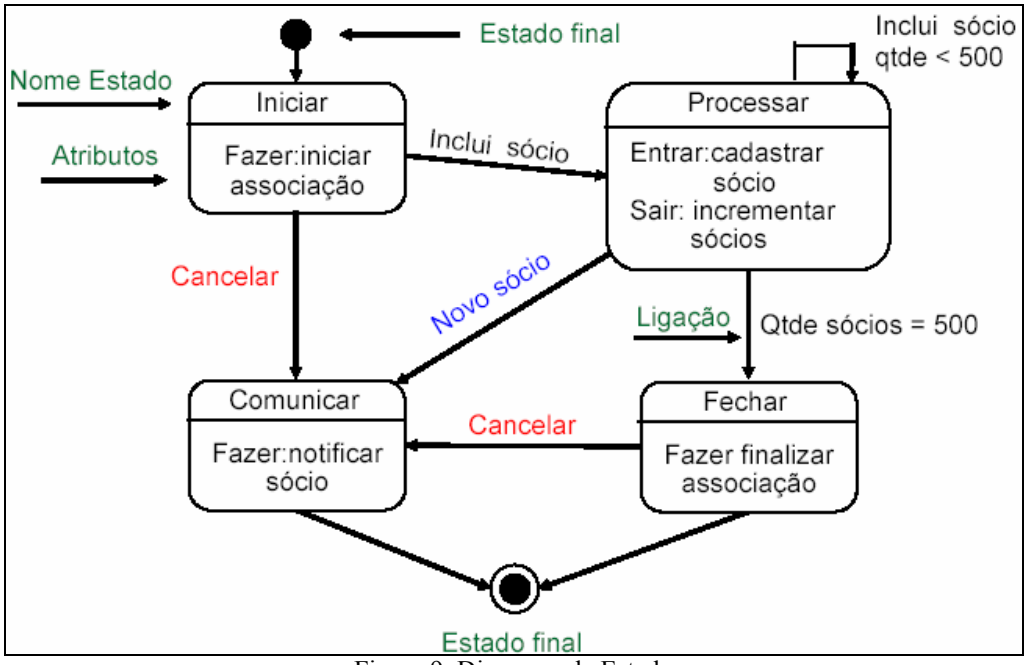


Figura 9. Diagrama de Estado
Fonte: CALDAS, P. O. (2009)

O diagrama de seqüência, Figura 10, é usado para modelar a interação entre objetos em um sistema. Ele normalmente é utilizado para a captura do comportamento de um único caso de uso. O diagrama representa os objetos e as mensagens que são passadas entre estes objetos dentro do caso de uso (ALMEIDA; DAROLT, 2001).

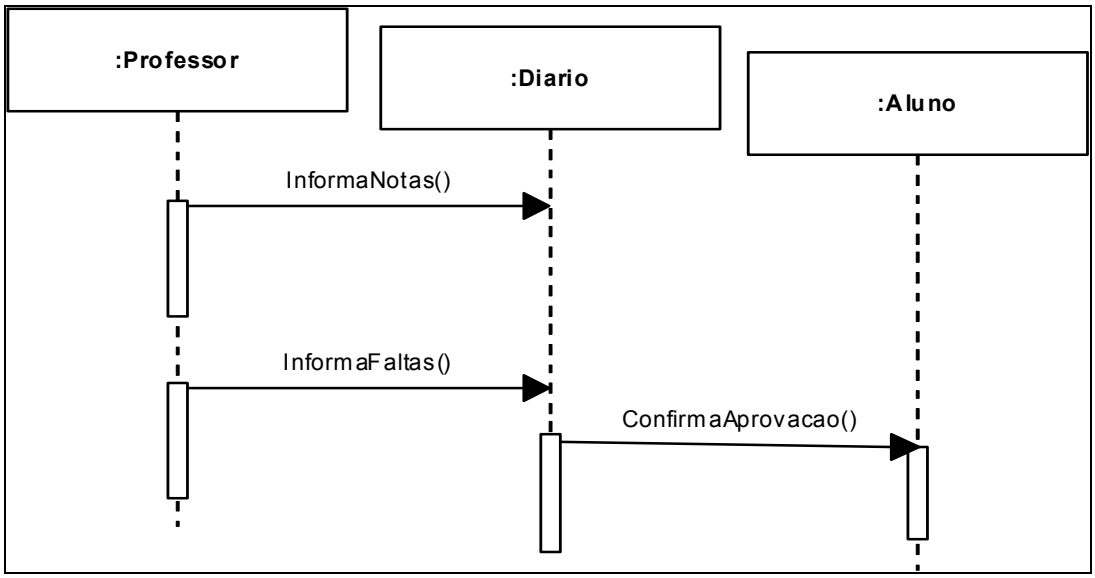


Figura 10. Diagrama de Seqüência
Fonte: Adaptado de ALMEIDA, A.; DAROLT, R. (2001)

O diagrama de colaboração, Figura 11, mostra de uma forma semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos. Normalmente escolhe-se entre usar o diagrama de colaboração ou o diagrama de seqüência. O diagrama de seqüência focaliza na seqüência cronológica do cenário que está sendo modelado, já o diagrama de colaboração focaliza no relacionamento entre os objetos e na compreensão dos efeitos sobre um objeto durante um cenário. Os objetos estão relacionados por associações e cada associação representa uma instância de associação entre as classes envolvidas. As associações mostram as mensagens enviadas entre os objetos, e a seqüência destas mensagens é determinada usando-se números seqüenciais (ALMEIDA; DAROLT, 2001).

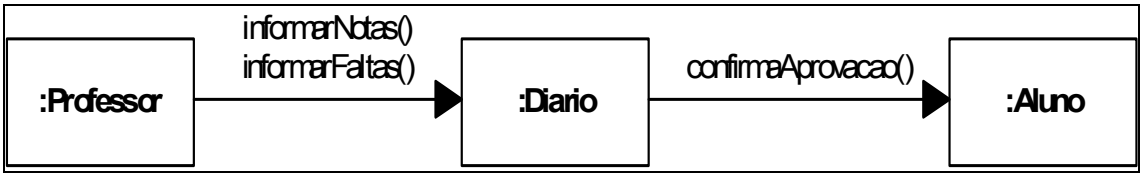


Figura 11. Diagrama de Colaboração
Fonte: Adaptado de ALMEIDA, A.; DAROLT, R. (2001)

O diagrama de atividades apresenta uma seqüência de atividades que começa em um ponto inicial e pode ter vários finais. O propósito é detalhar as ações e decisões em um processo (ALMEIDA; DAROLT, 2001). Este diagrama é maneira alternativa de se mostrar interações, com a possibilidade de expressar como as ações são executadas, o que elas fazem (mudanças dos estados dos objetos), quando elas são executadas (seqüência das ações), e onde elas acontecem (RUMBAUGH; JACOBSON, 2000).

O diagrama de atividade, Figura 12, representa um fluxo seqüencial de atividades, é normalmente utilizado para demonstrar as atividades executadas por uma operação específica. São estados de ação que contém a especificação de uma atividade a ser desempenhada por uma operação do software. Decisões e condições, como execução paralela, também podem ser mostrados na diagrama de atividade, além de poder conter especificações de mensagens enviadas e recebidas (RUMBAUGH; JACOBSON, 2000).

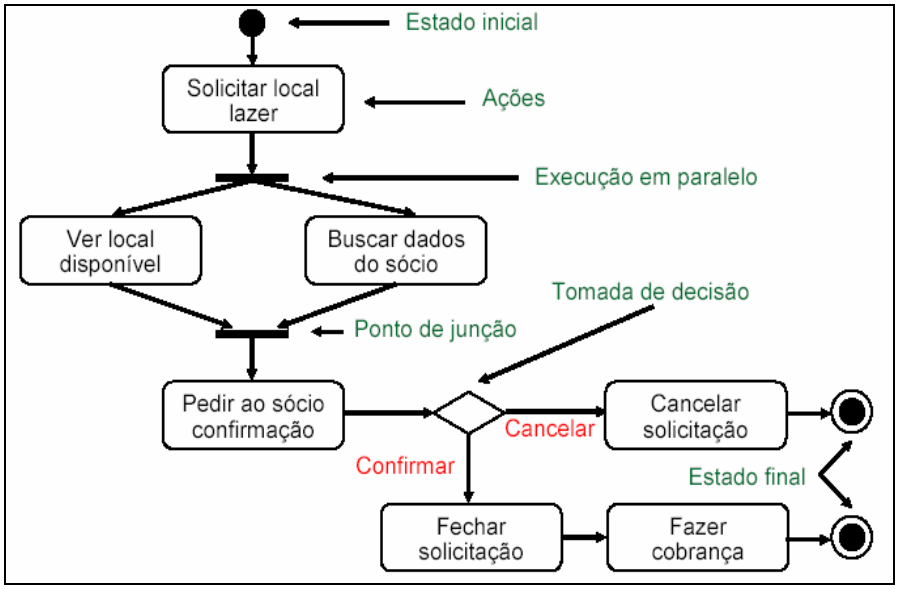


Figura 12. Diagrama de Atividade
Fonte: CALDAS, P.O. (2009)

Os diagramas de componentes, Figura 13, mostram o sistema por um lado funcional, apresentando as relações entre os componentes e a organização. Ele descreve os componentes de software e seus relacionamentos, representando a estrutura do código fonte ou protocolo. Os componentes desenvolvem na arquitetura física os conceitos e funções definidas na arquitetura lógica (classes, objetos e seus relacionamentos) (RUMBAUGH; JACOBSON, 2000). Estes diagramas podem ser utilizados para: modelar a organização do código fonte, modelar lançamento de executáveis, modelar fisicamente o banco de dados, modelar softwares adaptativos, engenharia de produção e engenharia reversa (ALMEIDA; DAROLT, 2001).

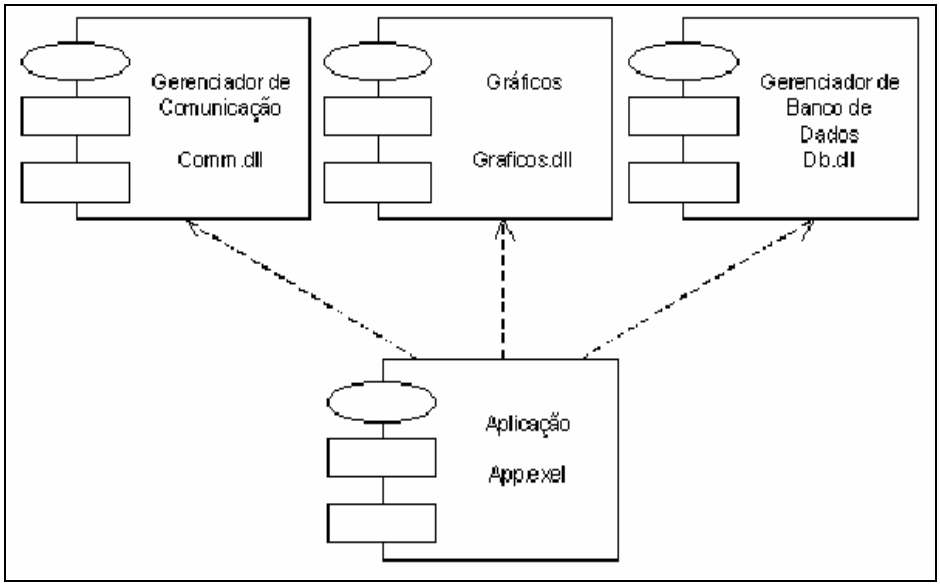


Figura 13. Diagrama de Componentes
Fonte: RUMBAUGH, J.; JACOBSON, I. (2000)

O diagrama de implantação, Figura 14, apresenta a arquitetura física do hardware e do software no conjunto. Eles são empregados para modelar a visão estática de implantação do software. Essa visão direciona primariamente a distribuição, entrega e instalação das partes que formam o sistema físico. Os diagramas de implantação são basicamente diagramas de

classes que focalizam os nós do sistema (ALMEIDA; DAROLT, 2001). É a última descrição física da topologia do sistema, descrevendo a estrutura de hardware e software que executam em cada unidade (RUMBAUGH; JACOBSON, 2000).

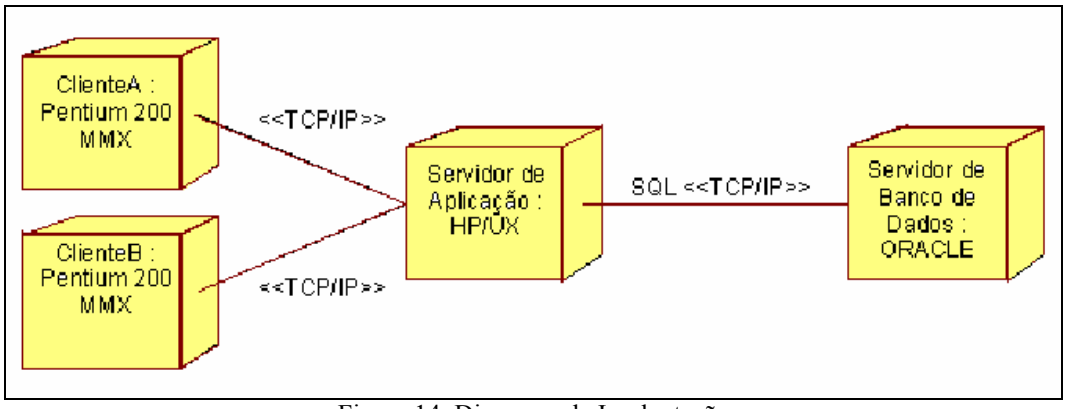


Figura 14. Diagrama de Implantação
Fonte: ALMEIDA, A.; DAROLT, R. (2001)

A utilização da notação UML agrega grande valor ao PS, visto que, seu conjunto de diagramas abrange varias fases do PS, simplificando e tornando visual as funções e restrições do software. Os diagramas UML devem ser usados na ER para favorecer o entendimento do cliente já que esta fase possui muitas incertezas, ambigüidades e omissões que contribuem para um escopo de requisitos fraco e com problemas. Eles contribuem também para compreensão do desenvolvedor sobre o problema e como criar o software.

3.3.5 Dicionário de Dados

O dicionário de dados forma uma lista organizada de todos os dados que se relacionam com o software. Sem o dicionário de dados diagrama de classes não pode ser considerado completo, pois este descreve entradas, saídas, composição de depósitos de dados e alguns cálculos intermédios. Ele é um ponto de referência de todos dados envolvidos,

permitindo associar um significado a cada termo utilizado. Esta ferramenta deve ser usada como parte da modelagem dos requisitos, pois ela torna-se possível a criação do banco de dados (ROCHA, 2004).

No dicionário de dados devem estar presentes os dados elementares, aqueles que não podem mais decompor, que fazem parte da entidade. As entradas de dados no dicionário de requisitos têm um identificador e descrição, que inclui (ROCHA, 2004):

a) o seu significado;

b) o seu conteúdo (só para dados compostos);

c) os valores permitidos e unidades (só para dados elementares);

d) a chave primária (só para depósitos de dados).

O Figura 15 é um modelo de dicionário de requisitos proposto em um quadro onde as informações necessárias estão de forma estruturada.

Tabela		Pedido Cabeçalho				
Descrição		Contém informações gerais do pedido de venda				
Chaves		Atributo	Domínio			Comentários
PK	FK		Tipo	Nulo	Condições	Valor Inicial	
×		Id_Pedido	Int	NN	Sem sinal		Código do pedido (seqüencial de *1* a *n*)
		Data_Emissão	Date	NN	dd/mm/aaaa	Now	Data de emissão do pedido
		Valor_Pedido	Num(12,4)	NN	0,00		Valor total do pedido
		Situação	INT	NN	Sem sinal 1- Aberto 2- Fechado 3- Cancelado 4- Parcial 5- Bloqueado 6- Jurídico	1	Situação em que se encontra o pedido
	×	Id_Vendedor	INT	NN	Sem sinal		Código do vendedor
	×	Id_Cliente	INT	NN	Sem sinal		Código do Cliente
	×	Id_Transportador	INT	NN	Sem sinal		Código do Transportador
	×	Id_Cond_Pagto	INT	NN	Sem sinal		Código das condições de pagamento

Figura15: Quadro de Dicionário de Dados
Fonte: CALDAS, P. O. (2009)

Descrevendo as informações, têm-se:

a) *tabela*: o nome da tabela correspondente;
b) *descrição*: sobre quais informações o quadro de dicionário de dados vai identificar;
c) *chaves*: a chave primária (PK) corresponde a identificação da tabela e a chave estrangeira (FK) corresponde as chaves das tabelas que compõe a tabela em questão;
d) *atributos*: variáveis que compõe a tabela;
e) *tipo*: identifica a qual tipo de informação pertence o atributo;
f) *nulo*: identifica se o campo pode ser nulo ou não;
g) *condições*: identifica quais as valores que podem povoar aqueles campos.
h) *valor inicial*: identifica qual o valor inicial do campo;

comentários: utilizado para incluir informações adicionais.

3.3.6 Etnografia

A etnografia é uma técnica de levantamento de requisitos que insere os engenheiros de requisitos na rotina de trabalho da organização tentando entender e descrever as principais atividades que são realizadas pelos usuários, pode ser vista na Figura 16. Na observação devem ser identificadas quais as atividades podem ser automatizadas, quem são os potenciais usuários, quais tarefas eles querem realizar com a ajuda do novo sistema (SOMMERVILLE, 2003).

A etnografia mostra que o estudo do campo de trabalho real é muito mais rico e complexo que aquele sugerido por simples modelos de software, pois as pessoas geralmente

acham difícil explicar detalhes de seu trabalho, e fatores sociais e organizacionais importantes podem ser observados (SOMMERVILLE, 2003).

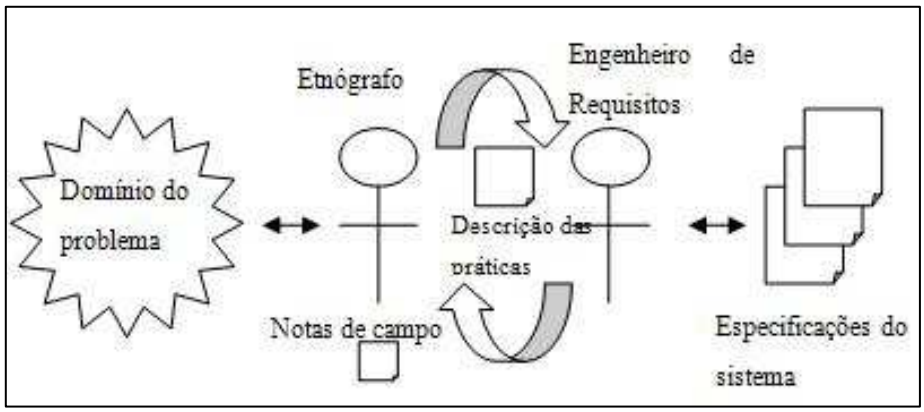


Figura 16. Etnografia
Fonte: VILLER, S.; Sommerville, I. (1999)

Apesar de a etnografia ser importante para o levantamento de requisitos, existem alguns riscos que dificultam o seu uso e devem ser considerados quando for implementar esta técnica (VILLER; SOMMERVILLE, 2000):

a) processo longo, pode tornar-se inaceitável para os cronogramas do PDS;
b) podem produzir resultados muito descritivos difíceis de serem compreendidos pelos engenheiros de requisitos;
c) realizar generalizações torna-se difícil a partir de dados situacionais;
d) a pesquisa de campo não segue um padrão, pode tornar o sucesso do método dependente das habilidades individuais do etnógrafo.

3.3.7 Prototipação

Um protótipo é uma versão inicial de um software, e pode ser utilizado para aquisição de experiência sobre o domínio. As organizações utilizam a prototipagem para gerar aos clientes executáveis das funcionalidades coletadas até o momento, podendo o cliente testá-las e indicar os pontos fortes e fracos do sistema, além de descobrir novos requisitos (SOMMERVILLE, 2003).

Como visto, a prototipagem é um boa pratica de se obter e analisar requisitos rapidamente, pode ser dividi-la em dois tipos (SOMMERVILLE, 2003):

a) a prototipagem *throw-away*: cria protótipos dos requisitos que geram maiores dificuldades e os que estão menos explícitos, a fim de achar novos requisitos e/ou desenvolve-los;
b) a prototipagem evolucionária: este tipo cria protótipos a fim de, gerar para os clientes executáveis úteis para o trabalho. Neste caso, os protótipos serão gerados a partir dos requisitos mais conhecidos para não proporcionar muitos problemas. Somente após a utilização extensiva deste protótipo inicial que os requisitos mal compreendidos devem ser implementados.

A prototipagem pode proporcionar benefícios como:

a) os clientes podem experimentar e descobrir o que é que eles realmente necessitam como suporte ao seu trabalho;
b) produção de protótipos em fases e a utilidade antes que riscos tenham ocorrido em custos elevados;
c) são essenciais para desenvolver a interface do usuário;

d) podem ser utilizados para teste do software e desenvolvimento da documentação;
e) gera um estudo detalhado dos requisitos que podem revelar inconsistências e omissões.

3.4 ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS

Segundo Turine e Masiero (2003) a análise de requisitos pode ser considerada uma tarefa fundamental para o desenvolvimento de software. O objetivo da análise é avaliar e revisar o escopo do software produzido pelo Levantamento de Requisitos, através do processo de descoberta, negociação e especificação. De acordo com Pressman (1995) nessa fase o engenheiro de requisitos especifica as funções e o desempenho que o software deve obter, indica a interface do software com outros sistemas e estabelece as limitações do PS.

A Análise de Requisitos proporciona a integração e melhoramento dos requisitos levantados, e a realimentação de requisitos no escopo, por causa do surgimento de novas necessidades durante sua realização. A análise deve separar os requisitos em grupos de semelhanças e criar um identificador único para cada requisito para serem identificados com maior agilidade (TURINE; MASIERO, 2003).

O engenheiro de requisitos, durante a fase de análise, deve encontrar e solucionar as inconsistências como os requisitos conflitantes, e funcionalidades incorretas geradas no escopo inicial. Após esse refinamento ele deverá decidir, junto ao cliente, o que será transformado em código, pois o documento gerado por está fase servirá de base para a ES nas fases posteriores (TURINE; MASIERO, 2003).

A Especificação dos Requisitos deve ser uma descrição sistemática e abstrata do que o software deve realizar, a partir do produto da análise. Ela vai proporcionar a solução de problemas definidos na análise e serão dispostos por um software computacional, ou seja, descreve as propriedades funcionais necessárias para resolver o problema do domínio e como a implementação poderá ser testada e verificada pelo usuário (PAULA FILHO, 2001).

Segundo Paula Filho (2001) a especificação é escrita por analistas em termos técnicos apropriados para o desenvolvimento do software e serve de comunicação entre análise e projeto. Requisitos especificados incorretamente, produzem o re-trabalho dos desenvolvedores e conseqüentemente atrasos no projeto e aumento dos custos, sendo responsável por uma parte significativa dos erros detectados ao longo do processo de desenvolvimento de sistemas, principalmente no caso de sistemas dedicados, de tempo real e críticos (LUTZ, 1993).

Paula Filho (2001) define que a Especificação de Requisitos deve possuir os seguintes aspectos: ser clara e não-ambígua, completa, correta, ser de fácil compreensão, consistente, concisa e confiável. Pressman (2006) acrescenta que nessa fase deve-se utilizar a Implantação da Função de Qualidade (IFQ), que é uma técnica para traduzir as necessidades do cliente, classificando os requisitos quanto à importância de cada um deles. Os três tipos de classificação são:

a) **requisitos normais:** são as necessidades estabelecidas pelo cliente (atendimento obrigatório);
b) **requisitos esperados:** requisito implícito no domínio ou em outros softwares similares, que o cliente imagina ser obvio demais (atendimento fortemente recomendado);

c) **requisitos excitantes:** requisitos que estão além do que se espera (seria bom atender).

Uma especificação de requisitos é completa se contém todos os requisitos importantes, definindo todas as definições de respostas do software para todas as classes imagináveis de entrada de dados em todas as situações imagináveis (FERNANDES, 2006). Sendo assim a linguagem natural pode oferecer vários riscos como ambigüidade, duplicidade e falta de modularização. Esses problemas podem ser solucionados com o uso de uma linguagem natural estruturada, ou seja, o analista cria um formulário padrão ou um *template* e utiliza na especificação. Porém esse formulário deve incluir:

a) descrever a função ou entidade;
b) descrever entradas e origens;
c) descrever saídas e destinos;
d) se utilizar outras entidades, indicar;
e) se existir pré e pós condição, indicar;
f) se existir efeitos colaterais, indicar.

O uso da linguagem natural estruturada, Quadro 1, pode ser uma boa técnica para tornar mais completo o escopo de requisitos e melhorar o entendimento dos requisitos no contexto por parte do desenvolvedor. A linguagem de descrição de projeto é uma espécie de linguagem de programação ou um algoritmo e também pode ser utilizada para realizar a especificação, porém ela é difícil de ser compreendida e realizada.

Função	Adicionar nós.
Descrição	Adiciona um nó em um desenho existente. O usuário seleciona o tipo de nó e seu posicionamento. Quando Adicionado ao desenho, o nó se torna a seleção atual. O usuário escolhe a posição do nó movimentando o cursor para a área que em que o nó será adicionado.
Entradas	Tipo de nó, Posição do nó, identificador do desenho.
Origem	Tipo de nó e posição do nó são entradas fornecidas pelo usuário; Identificador de desenho se origina da base de dados.
Saídas	Identificador do desenho.
Destino	O banco de dados do desenho. O desenho é designado para a base de dados, no término da operação.
Requer	Gráfico de desenho associado identificador de desenho de entrada.
Pré condição	O desenho é aberto e exibido na tabela do usuário.
Pós condição	O desenho é imutável, a não ser pela adição de um nó do tipo especificado em dada posição.
Efeitos Colaterais	Nenhum.

Quadro 1: Especificação de Requisitos
Fonte: Sommerville, I. (2003)

3.4.1 Negociação de Requisitos

A Negociação de Requisitos é uma das fases iniciais do PDS, mas o seu impacto reflete-se nas demais fases do desenvolvimento (RAMIRES, 2004). Lewicki e Litterer (1985) definem a negociação como o processo básico usado para resolver conflitos. Um requisito definido por um *stakeholder* pode não ser aceito por outro ou os requisitos podem estar além das limitações de recursos e devem ser revistos.

O processo de decisão na negociação deve encontrar uma alternativa (ponto de intersecção) dentro de um conjunto de alternativas aceitáveis pelos *stakeholders* e que seja viável. Utilizando uma abordagem interativa os requisitos são eliminados, combinados e/ou modificados de modo que cada parte alcance um grau de satisfação (PRESSMAN, 2006).

A Análise e Negociação de Requisitos estão ligadas e devem fazer parte de todos PDS. De acordo com Sommerville e Sawyer (1997) o processo deve ser seguido até à satisfação de todas as partes envolvidas, sendo um processo influenciado não só pela lógica e

argumentos técnicos, mas também pela política da organização e pela personalidade dos *stakeholders*. E assim obter um conjunto de requisitos que tenha a aprovação de todos, podendo haver mudanças ao longo do tempo. Por outro lado é essencial, do ponto de vista de gestão, controlar os requisitos considerados críticos (requisitos que se não forem satisfeitos comprometem a existência de todo o sistema) (GILB, 2003).

3.5 VALIDAÇÃO DE REQUISITOS

A especificação fornece uma visão do projeto aos analistas para que esses requisitos sejam direcionados aos desenvolvedores sendo submetidos a uma validação. Segundo Pfleeger (2004) a validação vai garantir que os requisitos atenderão as necessidades dos clientes.

Sommerville (2003) identifica que a validação deve trabalhar com o esboço completo do documento de requisitos, enquanto a análise trabalha com os requisitos ainda incompletos. Na Figura 17 podemos visualizar como estão dispostos às entradas e saídas de um processo de validação.

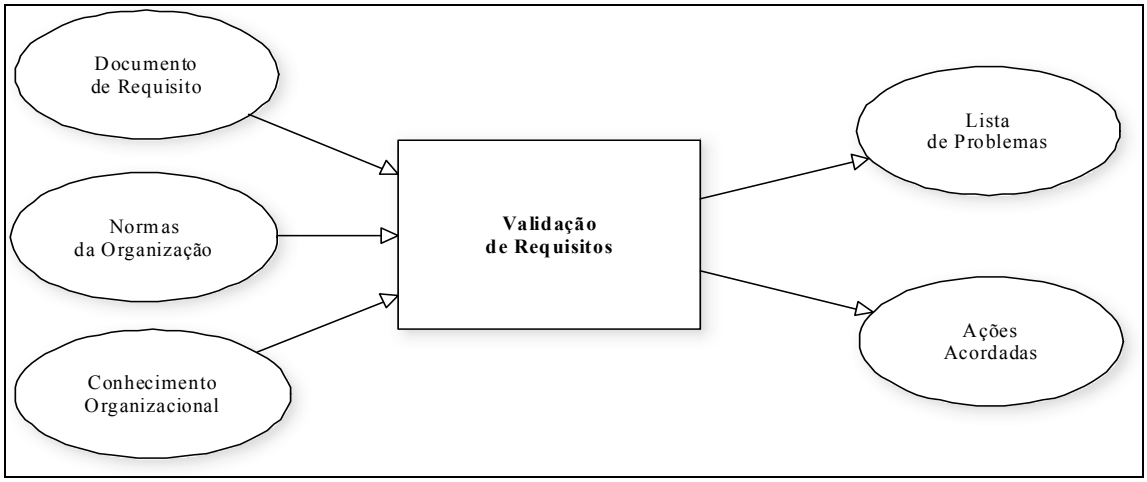


Figura 17. Processo de Validação de Requisitos
Fonte: SOMMERVILLE, I (2003)

A validação busca verificar e corrigir os problemas como: não conformidade com normas de qualidade, requisitos mal descritos, ambíguos, erros na modelação do sistema ou problema conflitos entre requisitos que não foram detectados no processo de Análise de Requisitos. Estes erros podem ocasionar grandes custos relacionados ao re-trabalho quando descobertos nas fases subseqüentes do PDS ou depois que o sistema estiver em operação, algumas das técnicas utilizadas para a validação podem ser vistas abaixo (SOMMERVILLE, 2003):

a) *revisões de requisitos:* os requisitos são revisados sistematicamente por uma equipe de revisores;
b) *prototipação:* um modelo executável é gerado e apresentado ao cliente para que ele possa experimentar e analisar se a suas necessidades estão sendo satisfeitas;
c) *geração de casos de teste:* casos de testes são criados para verificar se os requisitos são testáveis, caso seja difícil ou não se consiga testa-los, isso identificará sua difícil implementação e devem ser reconsiderados;
d) *análise automatizada da consistência:* ferramentas CASE são utilizadas para verificar a consistência dos requisitos, utilizando-se as regras dos métodos ou notação para analisar todos os requisitos.

3.6 GERÊNCIA DE REQUISITOS

A Gerência de Requisitos envolve a utilização de técnicas e ferramentas para gerenciamento de configuração e controle de versão, além de verificar inconsistências nas especificações, conforme os requisitos evoluem (HAZAN; LEITE, 2003). Segundo Leite, Rocha, Maldonado e Weber (2001) o processo de construção de software será cada vez mais baseado no conceito de evolução. Percebe-se que o próprio processo de definição de requisitos gera um *feedback* que acaba modificando os próprios requisitos, tornando inevitável as alterações nos requisitos.

A Gerência de Requisitos tem como objetivo principal controlar a evolução dos requisitos, seja para incluir novas necessidades, por causa de mudanças no ambiente em que o software será utilizado ou por se entender melhor o domínio do problema (HAZAN; LEITE, 2003). O Quadro 2 apresenta um conjunto de atividades de um processo de Gerência de Requisitos. Estas atividades apóiam a identificação, controle e rastreamento dos requisitos, bem como o tratamento das mudanças nos requisitos. As alterações podem ocorrer no desenvolvimento do software onde os custos e prazos do projeto serão comprometidos ou após a entrega do produto através de manutenções (BOEHM; PAPACCIO, 1988).

Atividades	Descrição
Receber as solicitações de alteração de requisitos	A equipe de ER recebe as solicitações de alteração de requisitos, ou por formulário padronizado, ou por meio de um sistema de solicitação de demandas.
Registrar novos requisitos	Novos requisitos devem ser recebidos formalmente, seja por formulário padronizado, ou por meio de controle sistemático.
Analisar impacto da mudança de requisitos	Uma análise criteriosa deve ser conduzida para avaliar o impacto do requisito a ser alterado sobre cada um dos seus requisitos relacionados, os quais podem ser identificados por meio das matrizes de rastreabilidade. Caso o impacto seja significativo, os requisitos (analisado e relacionados) devem ser revistos.
Elaborar relatório de impacto	Deve ser mantido um histórico de alterações para cada requisito, permitindo uma visão cronológica das principais mudanças.
Notificar os envolvidos	Conjunto de pessoas para as quais pode haver um impacto devido à alteração de requisitos (alteração, inclusão ou exclusão de requisitos) e devem ser notificados.
Coletar métricas	As métricas devem ser utilizadas e coletadas periodicamente para o acompanhamento das atividades da Gerência de Requisitos.

Quadro 2. Atividades de um Processo de Gerência de Requisitos
Fonte: HAZAN, C.; LEITE, J.C.S.P. (2003)

O ambiente onde o software está situado pode sofrer evoluções freqüentes, o que exige modificações do software e nas regras de negócio referente ao domínio (ZOWGHI; OFFEN, 1997). Por isso torna-se necessário que a gerência adote um plano para diminuir o impacto das alterações sobre o PDS. Segundo Kotonya e Sommerville (1998) a tentativa de antecipar as mudanças dos requisitos instáveis é considerada uma boa prática da Gerência de Requisitos. Assim deve-se classificar os requisitos identificando os mais voláteis e analisar as possíveis mudanças que poderão ocorrer.

Gerenciar as mudanças de requisitos é fundamental para o sucesso do desenvolvimento, este é o principal risco que atinge 80% dos PDS (JONES, 1996). De acordo com Kotonya e Sommerville (1998) os requisitos não podem ser efetivamente gerenciados sem o rastreamento. A rastreabilidade do requisito deve identificar quem sugeriu o requisito, porque o requisito existe, a quais outros requisitos ele está relacionado e como ele está

relacionado com outras informações como artefatos de projeto, implementação e documentação de usuário (HAZAN; LEITE, 2003).

Segundo Sayão e Leite (2006) a rastreabilidade dos requisitos pode ser vista como a habilidade de acompanhar e descrever a vida de um requisito, em ambas as direções permitindo as alterações dos artefatos e possibilitando uma estimativa mais precisa do esforço e tempo necessários. O rastreamento também permite encontrar outros requisitos que podem ser afetados quando uma mudança é solicitada. O Quadro 3 apresenta possíveis razões para que ocorra mudança nos requisitos.

Atividades	Descrição
Erros em requisitos, Conflitos e inconsistências	Conforme os requisitos são analisados e implementados, erros e inconsistências surgem e devem ser corrigidas.
Evolução do conhecimento do cliente	Conforme os requisitos são desenvolvidos, clientes e usuários finais desenvolvem uma melhor compreensão do que desejam.
Problemas técnicos, de custo ou cronograma	Problemas podem ser encontrados na implementação dos requisitos. Pode ser muito custoso implementar certos requisitos.
Mudanças nas prioridades do cliente	As prioridades do cliente podem mudar durante o desenvolvimento do sistema como resultado de mudanças no ambiente de negócios.
Mudanças de ambiente	O ambiente no qual o software será instalado pode mudar, assim os requisitos devem ser modificados para manter compatibilidade.
Mudanças organizacionais	A Organização que pretende usar o software pode mudar sua estrutura e processos, resultando em novos requisitos de sistema.

Quadro 3. Fatores de Mudança de Requisitos
Fonte: KOTONYA, G.; SOMMERVILLE, I. (1998)

A alteração, inclusão e exclusão dos requisitos devem acontecer durante a realização da ER, em um período determinado pelo projetista para que as próximas fases do PDS não sejam interrompidas pela mudança no escopo.

Segundo El Emam, Höltje e Madhavji (1997) os principais problemas relacionados à Gerência de Requisitos são os seguintes:

a) dificuldades de descobrir, de forma consistente, as mudanças nos requisitos;
b) dificuldades para chegar a um consenso sobre as mudanças importantes para os *stakeholders*;
c) dificuldade em manter o documento de requisitos consistente;
d) dificuldade em estimar adequadamente os recursos necessários para implementar as mudanças nos requisitos.

3.6.1 Requisitos Estáveis e Voláteis

Os requisitos estáveis derivam da atividade principal do cliente e podem, também, ser derivados dos modelos de domínio. Os requisitos voláteis são aqueles que têm maiores possibilidades de mudança (SOMMERVILLE, 2003). Segundo Kotonya e Sommerville (1998) esses requisitos costumam ser específicos para instanciar um software em seu ambiente. A variação que os requisitos sofrem durante o ciclo de vida de um projeto é o que se chamamos de volatilidade de requisitos. A seguir, podemos ver um exemplo de requisitos estáveis e voláteis no Sistema Controle Universitário (CASTRO, 2009):

a) **requisitos estáveis:** cadastro de estudantes, cadastro de cursos;
b) **requisitos voláteis:** cadastro das turmas, grupo de cursos, tipo do curso (à distância ou presencial).

A volatilidade de um requisito é determinada pela influência de um fator externo sobre esse requisito. Quanto mais sensível um requisito é a um fator externo, mais volátil ele é (CASTRO, 2009). Um projeto que possui um grande número de requisitos voláteis vai gerar maior risco de não ser entregue no prazo acordado ou dentro da faixa de custos estabelecidos.

Existem diversos fatores que podem determinar se os requisitos de um projeto são voláteis ou não, abaixo podemos ver alguns (KOTONYA; SOMMERVILLE, 1998):

a) erros, conflitos e inconsistências nos requisitos;
b) evolução do conhecimento do sistema pelos clientes/utilizadores;
c) problemas técnicos, de planificação, ou de custo;
d) alteração das prioridades do cliente;
e) alterações ao contexto do sistema;
f) alterações na organização;

As alterações nos requisitos são inevitáveis, para isso os analistas devem trabalhar com modelos que se adequem a essa volatilidade. Como não se pode prever o que vai acontecer no futuro deve-se utilizar técnicas que ajudem a identificar os possíveis requisitos voláteis. Eles são classificados em quatro áreas (SOMMERVILLE, 2003):

a) **requisitos mutáveis:** requisitos que se modificam devido ao ambiente que o sistema vai operar;
b) **requisitos emergentes:** requisitos que surgem à medida que a compreensão do cliente do sistema se desenvolve;
c) **requisitos conseqüentes:** requisitos que resultam da introdução do sistema de computação;

d) **requisitos de compatibilidade:** requisitos que dependem de outros sistemas ou processos de negócios específicos dentro da organização.

A volatilidade histórica é uma técnica que pode ser utilizada para analisar o volume de requisitos que sofreram modificações. Ela consiste em levantar a volatilidade ocorrida em desenvolvimentos passados.

A volatilidade potencial também pode ser utilizada, ela está essencialmente ligada a fatores externos ao projeto no qual o requisito está inserido e que podem provocar alterações nos mesmos (CASTRO, 2009). Essas técnica visam combater os riscos na ER e produzir o se principal artefato que é o documento de requisito de forma consistente e completa.

3.7 DOCUMENTO DE REQUISITOS

O documento de requisitos é um documento formal utilizado para comunicar os requisitos aos clientes, engenheiros e gestores. A boa organização lógica do documento e a sua redação correta são condições essenciais para que ele se torne útil na prática**.** Seu conteúdo, originado da ER, deve definir quais funções são executadas sobre que dados, para produzir quais resultados em qual ambiente e os usuários. As pessoas envolvidas com o desenvolvimento (usuários, técnicos e engenheiros) devem participar da elaboração do documento, que deve definir corretamente todos os requisitos do software, descrevendo as necessidades do software, e não o processo de produção (FERNANDES, 2006).

Segundo Nuseibeh e Easterbrook (2000), a documentação dos requisitos de um sistema é de extrema importância, para garantir que os mesmos possam ser lidos, analisados e

validados. Sommerville (2003) define algumas características que o documento de requisitos deve ter, são elas:

a) especificar o desempenho do software externamente;
b) especificar as restrições ao software;
c) ser fácil a modificação;
d) ser adotada como ferramenta para a manutenção;
e) registrar a tática abordada pelo ciclo de vida;
f) identificar respostas aceitáveis para eventos indesejáveis.

O Documento de Requisitos deve incluir segundo Oliveira (2002) e Caldas (2009):

a) *introdução:* propósito, nome, objetivos, limites de atuação, benefícios esperados e visão geral;
b) *informações de suporte:* definições, glossário e referências;
c) *descrição geral do software:* perspectiva de produto (o ambiente que o software deverá ser executado: interconexões, interfaces, externas, operações, sistemas de comunicação, restrições de memória e requisitos da plataforma), características de usuários, restrições (definir obrigações ou limitações às opções de desenvolvimento, justificando), infra-estrutura técnica e administrativa, suposições e dependências e requisitos futuros;
d) *informações sobre viabilidade:* estudo realizado de viabilidade técnica, econômica e legal;
e) requisitos: levantamento, análise (requisitos de desempenho, requisitos de bases de dados lógicas, restrições de projeto), especificação (para cada uma

descrever: entradas válidas e inválidas, operações realizadas, resultados ou saídas aceitáveis e tratamento de exceções).

f) *modelagem:* construção de modelo lógico e físico*;*

g) *dicionário de dados:* construção de um quadro com as informações necessárias para modelagem do banco de dados;

h) *Interface:* construção de um modelos de interface;

i) *apêndices*.

Como o documento é direcionado, também, aos clientes torna-se importante relacionar o sistema com os objetivos do negócio da organização e a razão do negócio para o sistema. Assim é mais fácil o acesso aos mesmos e a confecção de novas versões do documento de requisitos (OLIVEIRA, 2002). Este documento deve ser gerado no inicio do ciclo de vida do software, servindo de base de informações para desenvolvimento das fases seguintes.

4 MODELOS DE CICLO DE VIDA DO SOFTWARE

Os modelos de ciclos de vida representam o PDS de uma forma abstrata, cada um de um jeito que lhe é peculiar. Os modelos descrevem os diferentes processos durante o desenvolvimento, que por sua vez estão divididas em ações que geram tarefas tornando o software gerenciável. Eles devem ser utilizados para alcançar as metas propostas pelo desenvolvimento, e gerar softwares de qualidade. Softwares de grande porte podem utilizar mais de um modelo dentro do PDS (SOMMERVILLE, 2003).

Os PS possuem características que são particulares, e cabe aos engenheiros de software definir qual o melhor modelo para ser utilizado no desenvolvimento e adaptá-lo se for necessário (PRESSMAN, 2006).

Gustafson (2003) identifica os modelos Cascata, Prototipação, Incremental, e Espiral como sendo os modelos mais conhecidos. Porém estes são modelos tradicionais que fazem parte de um contexto, onde o custo para que as modificações feitas era alto, devido às limitações dos computadores e falta de ferramentas modernas para apoiar a criação do software. Atualmente, as novas metodologias visam à rapidez no fornecimento de pedaços do software aliada à qualidade do processo e do produto. Assim surgiram os métodos ágeis, popularizados quando Kent Beck o criador do Manifesto Ágil, indicando alguns princípios comuns (VIANA; DESCHAMPS, 2008):

a) *indivíduos e interações:* ao invés de processos e ferramentas;

b) *software funcionando:* é mais significante do que documentação detalhada;

c) *colaboração do cliente:* mais eficaz que a negociação de contratos;

d) *adaptação a mudanças:* torna-se mais eficaz que um plano inicial.

A metodologia ágil proporciona um desenvolvimento que permite alterar constantemente o código sem diminuir a qualidade, visto que as mudanças estão presentes na vida do software. O modelo ágil mais conhecido é o *Extreme Programming (XP)*, mas outro modelo vem ganhando espaço no desenvolvimento ágil, o SCRUM. Posteriormente serão apresentados os modelos tradicionais e ágeis citados e suas características (VIANA; DESCHAMPS, 2008).

4.1 MODELO CASCATA OU CLÁSSICO

O Modelo Cascata, Figura 18, foi idealizado por Royce em 1970 e propõe um desenvolvimento de forma sistemática e seqüencial, sendo que, uma fase nova só começa após o término da anterior. O cliente participa ativamente do desenvolvimento, validando cada etapa concluída. Cada fase de desenvolvimento prossegue em uma ordem restrita, sem qualquer sobreposição ou passos iterativos (PRESSMAN, 2006). Mas na pratica essas fases se sobrepõe interagindo entre si.

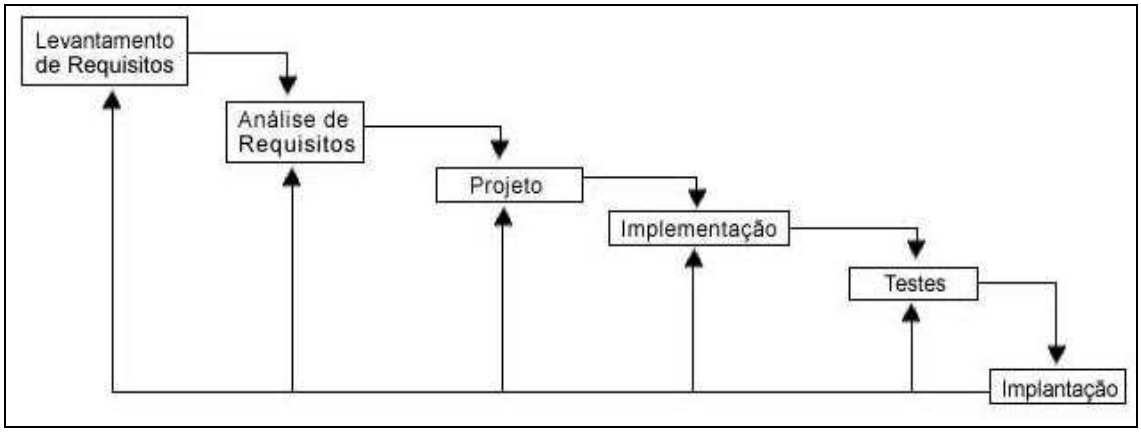


Figura 18. Modelo Cascata
Fonte: BEZERRA, E. (2002)

Assim, o desenvolvimento é efetuado de forma rígida, com pontos de controle bem definidos, facilitando a gerência de projetos. Porém, fases como a de requisitos, análise e

desenho necessitam estar bem claras junto às necessidades do cliente, para que não ocorram erros ou riscos, pois este modelo não permite flexibilidade e revisão. Uma vez na fase de testes se torna oneroso o retorno a fases iniciais, como a de requisitos, sendo desprezada novas funcionalidades requeridas pelo cliente e correções de riscos (SOMMERVILLE, 2003).

O modelo cascata é de baixa visibilidade para o cliente, que só recebe o resultado final do projeto e pode gerar incompletude dos requisitos, pois inserir novos requisitos em fases mais adiantadas pode ser inviável (SOMMERVILLE, 2003).

4.2 MODELO INCREMENTAL

O Modelo Incremental possui características para suprir os problemas de inflexibilidade do Modelo Cascata (BEZERRA, 2002). Inicialmente, o cliente define seus requisitos básicos, que formarão um conjunto ou incremento, que deverá ser desenvolvido de acordo com um PDS e entregue ao cliente, sendo o primeiro definido como o núcleo do produto (PRESSMAN, 2006).

A partir da entrega do primeiro incremento, o cliente poderá testar parte do software e definir outros requisitos para gerar outros incrementos, e assim sucessivamente até que se satisfaçam as necessidades (SOMMERVILLE, 2003).

O desenvolvimento por incrementos, Figura 19, permite que erros e riscos gerados pelos requisitos em fases anteriores possam ser solucionados em fases posteriores. Este modelo não necessita de planejamento criterioso e não tem um cronograma e previsão custo definidos, gerando pouco conhecimento sobre o software a ser desenvolvido, previamente (BEZERRA, 2002).

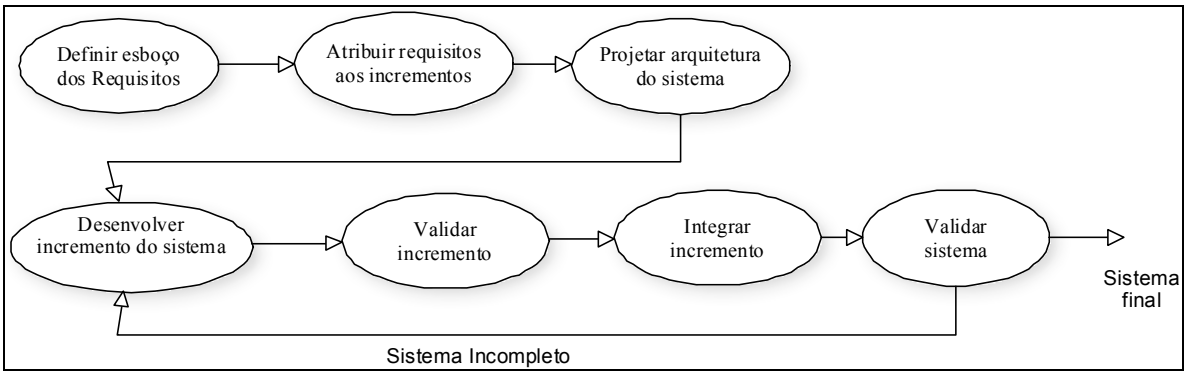


Figura 19. Modelo Incremental
Fonte: SOMMERVILLE, I. (2003)

4.3 MODELO PROTOTIPAGEM

O Modelo Prototipagem desenvolve uma versão descartável da aplicação, para o cliente testar conceitos e requisitos (GUSTAFSON, 2003). O protótipo deverá ser desenvolvido para elucidar problemas de detalhamento dos requisitos e para definir progressivamente as funcionalidades.

O primeiro protótipo busca definir os objetivos gerais da aplicação e os requisitos, que no momento, o cliente tem conhecimento. O desenvolvimento do protótipo segue as fases de codificação e testes do projeto, informalmente, sem considerar aspectos de qualidade e manutenção. O protótipo, depois de implantado e testado pelo cliente, ajuda o desenvolvedor a compreender o que se precisa construir. Baseado na avaliação e novos requisitos do cliente, os protótipos são alterados até que se atinja o produto desejado. Este PDS deve ser utilizado apenas para desenvolver softwares considerados de porte pequeno (PRESSMAN, 2006).

A prototipagem pode ser utilizada por engenheiros de software dentro de outros modelos de desenvolvimento, mesmo sendo um modelo completo, pois ajuda a formar requisitos mais consistentes. Após a concordância entre o protótipo gerado e as exigências do

cliente, o software é desenvolvido seguindo as mesmas fases do modelo cascata (GUSTAFSON, 2003).

4.4 MODELO ESPIRAL

O Modelo Espiral foi proposto por Boehm em 1988, e tinha o objetivo de juntar as qualidades do Modelo Cascata, aspectos de controle e desenvolvimento sistemático, com o Modelo Prototipagem, que é um modelo evolucionário e interativo com o cliente. A sua abordagem evolutiva em relação à ES permite ao desenvolvedor e ao cliente compreenderem e avaliarem os riscos de cada ciclo evolutivo na espiral. O Modelo Espiral tem como característica o tratamento dos riscos de forma explicita, diferente dos outros modelos. Os riscos são circunstâncias adversas que podem surgir durante o PDS impedindo o andamento do processo ou reduzindo a qualidade do produto (SOMMERVILLE, 2003).

No Modelo Espiral o desenvolvimento tem um formato cíclico, e possui um processo interativo que tende a melhorar o protótipo aos poucos, enquanto diminui os riscos (IEEE, 1998). Este modelo busca melhorar o desenvolvimento das versões (protótipos completos) a cada volta na espiral até chegar ao produto final, requerido pelo cliente.

O Modelo Espiral mostra-se mais flexível e têm maior adaptação as características de cada projeto. O risco é analisado a cada evolução do desenvolvimento, tornado cada versão mais confiável. O Modelo Espiral organiza o desenvolvimento em conjuntos de quatro fases, que se sucedem até se obter o software final (SOMMERVILLE, 2003):

a) *definição dos objetivos:* inicio com a determinação dos objetivos, alternativas e restrições;

b) *análise de riscos:* análise e avaliação de alternativas, identificação e solução de riscos, se o risco for considerado inaceitável o projeto pode ser interrompido;

c) *engenharia:* escolha do modelo de desenvolvimento e o desenvolvimento do produto;

d) *planejamento:* o produto é avaliado pelo cliente e planeja-se a próxima volta no ciclo.

A identificação e gerenciamento de riscos é, atualmente, considerada uma atividade muito importante no desenvolvimento de software (SOMMERVILLE, 2003). No Modelo Espiral o software evolui à medida que cada processo avança na espiral, sendo que, o desenvolvedor e o cliente entendem melhor e reagem aos riscos de cada nível, conforme Figura 20. A prototipagem pode ser aplicada para minimizar os riscos de cada evolução da espiral. Com isso o Modelo Espiral consegue identificar e criar planos de contingência para os riscos com maior antecedência, diminuindo os efeitos negativos proporcionados por esse evento (PRESSMAN, 2006).

Contudo, o Modelo Espiral exige grande competência na avaliação dos riscos, para que se tenha sucesso neste PDS. O custo e cronograma do projeto devem ser ajustados conforme a avaliação e resposta do cliente após a entrega de cada protótipo. O Modelo Espiral oferece a organização desenvolvedora do software, a possibilidade do desenvolvimento em paralelo de múltiplas partes do projeto, cada uma sendo abordada de modo diferenciado, com isso torna-se necessário o uso de técnicas para sincronizar os cronogramas e determinar os indicadores de custo (PRESSMAN, 2006).

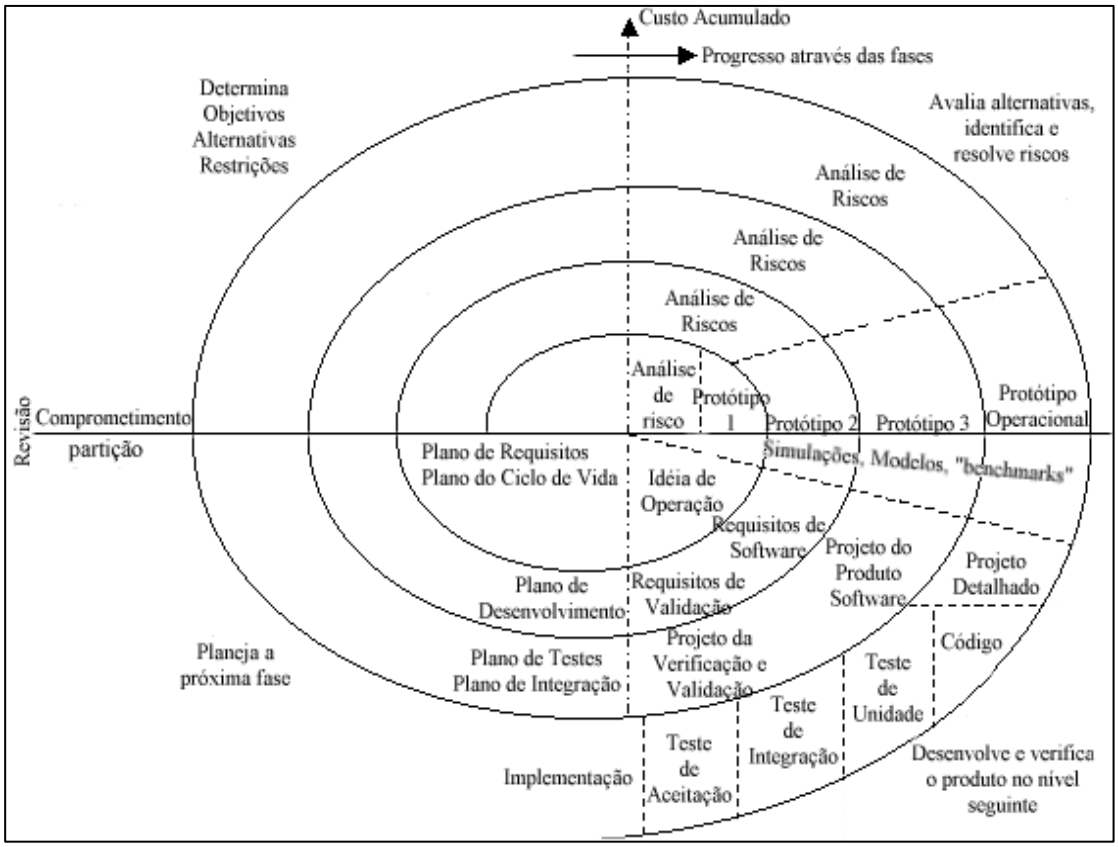


Figura 20. Modelo Espiral
Fonte: SOMMERVILLE, I. (2003)

4.5 MODELO *EXTREME PROGRAMMING*

A metodologia XP é um PDS criado no final da década de 1980, nos Estados Unidos, por Kent Beck. A programação, no XP, adota o desenvolvimento em pares de programadores e há a geração de versões operacionais a cada tarefa finalizada pela equipe de desenvolvimento (SCOTT, 2003). Está metodologia é ideal para ser usada em projetos em que o cliente não sabe exatamente o que deseja e pode mudar de opinião, varias vazes, durante o desenvolvimento do projeto (VIANA; DESCHAMPS, 2008).

Segundo Caetano (2004) o XP envolve um processo simples, divido em pequenas iterações que devem durar, relativamente, um curto espaço de tempo, onde os requisitos são levantados em contato direto com os clientes. Este deve permanecer ativo indicando

modificações e esclarecendo dúvidas da equipe. Cada iteração por sua vez, é submetida a todas as fases do ciclo de vida do projeto, além de testes. Ao final é realizada a reconstrução do projeto à medida que novas funcionalidades são incrementadas ao software (VASCONCELOS, 2005). A metodologia ainda oferece a prática da refatoração que permite a melhoria contínua da programação. Os desenvolvedores reestruturam o sistema sem alterar seu comportamento, para remover duplicação, melhorar a comunicação, simplificar ou adicionar flexibilidade (VIANA; DESCHAMPS, 2008).

Porém, o XP não oferece uma avaliação dos riscos, mesmo os requisitos sendo captados de modo informal e estando sujeito a ambigüidades e interpretações erradas. Outra característica do XP é a pouca ou nenhuma documentação dos processos que dificulta o entendimento e a identificação dos problemas (VIANA; DESCHAMPS, 2008). O planejamento do software, referente a cronogramas e custos, fica vulnerável pelo fato do software sofrer diversas modificações ao longo do seu desenvolvimento.

4.6 MODELO *SCRUM*

A metodologia ágil *Scrum* criada por Jeff Sutherland, Ken Schwaber e John Scumniotales na década de 1990 e procura um desenvolvimento iterativo e incremental. A metodologia *Scrum* objetiva o aumenta da produtividade e a redução do tempo para obtenção de resultados. Com isso, ela facilita a adaptação a processos de pouca experiência de desenvolvimento, sendo um modelo relativamente simples (ROCHA; BELCHIOR, 2006).

A metodologia tem inicio com o processo de Planejamento da *Sprint* (*Sprint Plannig Meeting*) que consiste em uma reunião onde são feitas estimativas que procurarão estabelecer os itens que serão executados em uma *sprint*. No *Scrum* a iteração é chamada de

sprint e, um projeto é composto por vários *sprints* do mesmo tamanho, que desenvolvem o *Product Backlog* que é uma lista das funcionalidades, os requisitos funcionais. Consequentemente é produzido o *Impediments Backlog,* que contém todos os itens que impedem o progresso do projeto e geralmente estão associados a riscos (SUTHERLAND, 2001).

O *Scrum* divide o desenvolvimento em *sprints* de 30 dias e todo o dia ocorre uma reunião, de 15 minutos, onde cada membro da equipe dará as suas impressões a respeito do projeto e o gerente de projeto verifica o andamento do trabalho, observando problemas, verificando a continuidade do processo, resolvendo mal-entendidos (ROCHA; BELCHIOR, 2006).

A metodologia *Scrum* é semelhante a do XP, pois utiliza equipes pequenas e requisitos instáveis ou desconhecidos, oferecendo a visibilidade do andamento do projeto e a previsibilidade dos acontecimentos, sendo diferentes apenas nas dimensões (ROCHA; BELCHIOR, 2006).

O *Scrum* como o *XP* não tem uma fase de requisitos bem definida, sendo que o software sofrerá diversas interações até que satisfaça o cliente. Com isso, o planejamento de custos e cronogramas fica comprometido e a manutenção também, pois há pouca produção de documentos. Nesse contexto seria difícil implementar uma gerência de riscos visto que as fases são bastante curtas e os resultados tem que ser rápidos, independente da qualidade do produto gerado.

5 GERENCIAMENTO DE RISCOS

Os riscos na área de software foram representados de forma sistemática por Barry Boehm, nos anos 80, com o desenvolvimento do modelo em Espiral. Trata-se de um modelo de desenvolvimento iterativo e dirigido a riscos, pois a cada iteração é feita uma análise para se conhecer os riscos (BOEHM, 1988).

Porém, a área que trata riscos na ES evoluiu, passando de uma análise dentro do modelo de desenvolvimento, como era apresentada pelo Modelo Espiral, para se tornar uma gerência que deve percorrer todos os processos do ciclo de vida de software. O desenvolvimento de software, geralmente, possui incertezas que podem causar atrasos de cronogramas, aumento de custos ou produtos que não satisfaçam os clientes, além de preocupações, dúvidas e desconhecimento (SOMMERVILLE, 2003).

O sucesso PDS pode ser afetado por um ou mais eventos, capazes de causar impactos negativos ao software. A gerência de riscos tem a tarefa antecipar estes eventos no desenvolvimento do software com objetivo minimizar os impactos através da identificação de riscos, criação de planos de contingência e o controle dos riscos evitando o insucesso do software (HIGUERAG, 1994).

O Gerenciamento de Riscos (GR) proporciona a ES, o aumento da qualidade no desenvolvimento de software e, conseqüentemente, a do produto final. O mercado está exigindo cada vez mais das empresas produtoras de software decisões rápidas, melhor alocação de recursos e metas bem traçadas, que se tornam pré-requisitos para o sucesso de PS. O sucesso de um PS está interligado ao sucesso da própria organização, ou seja, o sucesso do seu produto determina o sucesso de toda a organização (GUSMÃO; MOURA, 2004).

Segundo Keil (1998) uma das causas para a alta taxa de falha em PS é que os gerentes não estão tomando as medidas prudentes necessárias para avaliar e gerenciar os riscos envolvidos nesses projetos. Os riscos devem ser identificados e classificados de tal forma que possam ser sugeridas estratégias adequadas para minimização dos mesmos.

A GR é uma técnica eficiente no combate à falhas no desenvolvimento do software, agindo de forma pró-ativa, ou seja, tratando deles quando são apenas riscos (GUSMÃO; MOURA, 2004).

5.1 ATIVIDADES DA GERÊNCIA DE RISCO

Os guias de GR e modelos possuem um consenso entre as atividades fundamentais deste processo, sendo que todas as atividades devem ser baseadas e centradas na comunicação entre todas as partes do PS, e serem realizadas de forma cíclica e contínua dentro do processo de gerenciamento. As atividades comuns que formam a GR são (GUSMÃO; MOURA, 2004):

a) **planejar a gerência de risco:** definir a tática da GR, os recursos necessários e, as efetivação das ações necessárias;
b) **identificar riscos:** levantar todas as possibilidades de riscos existentes;
c) **analisar riscos:** buscar os aspectos mais importantes de cada risco, eles devem ser categorizados e priorizados;
d) **planejar respostas aos riscos:** traçar planos de ação para combate aos riscos e de contingência para os riscos que se estão além das capacidades da GR;
e) **monitorar riscos:** observar a efetividade dos planos de ação;

f) **controlar riscos:** avaliar a situação corrente e se necessário alterar as táticas de mitigação, utilizar as ações de contingência, encerrar os trabalhos relacionados a um risco, quando este não mais existir;

g) **comunicar os riscos:** a comunicação entre as equipes e membros do projeto de software.

5.2 IDENTIFICAÇÃO DE RISCOS

Segundo Carr (1993) os riscos em um projeto podem ser classificados em: internos ou externos; conhecidos ou desconhecidos. Os riscos internos são aqueles sobre os quais o gerente do projeto tem controle e pode reduzi-los por meio de ações. Os riscos externos são aqueles sobre as quais o gerente do projeto tem pouco ou nenhum controle e estão associadas a estimativas ou hipóteses externas.

Os riscos conhecidos podem ser identificados e analisados para formar planos de combate, estes serão os riscos abordados neste trabalho. Já os riscos desconhecidos não podem ser gerenciados, sendo que os gerentes de projeto podem endereçá-los para um plano de contingência baseada na experiência em projetos realizados anteriormente. A identificação dos riscos pode ser realizada com ajuda de técnicas elaboradas e recomendadas pelo guia *Project Management Body of Knowledge* (PMBOK) que são (PMI, 2004):

a) *revisão da documentação:* uma revisão na documentação do projeto para identificar problemas de inconsistência e qualidade nos planos de projeto, estes problemas podem, por sua vez, ser indicadores de risco de projeto;

b) *Brainstorming*: grande lista de possíveis riscos, normalmente realizada por especialistas que não fazem parte do desenvolvimento;

c) *entrevista:* uma das principais fontes de obtenção de dados, realizadas com os participantes experientes, além de especialistas;

d) *lista de verificação:* as informações passadas e a experiência adquirida em PS servem de base, normalmente complementam outras técnicas;

e) *técnica delphi:* consenso entre especialistas – um questionário é distribuído entre os especialistas e os mesmos os respondem anonimamente e um facilitador recolhe os questionários e o redistribui para que possíveis comentários sejam adicionados, após algumas repetições deste processo o consenso deve ser alcançado;

Procurando formar um escopo de riscos, Barki, Rivard e Talbot (1993) propuseram um instrumento de conhecimento, com 42 itens relativos a várias características de um PDS, conforme Quadro 4.

Aquisição tecnológica	**Intensidade de conflitos**
1. Necessidade de novo hardware 2. Necessidade de novo software 3. Grande número de fornecedores de hardware 4. Grande número de fornecedores de software	28. Grande intensidade de conflitos entre os membros da equipe 29. Grande intensidade de conflitos entre usuários e membros da equipe
Tamanho da aplicação	**Extensão de mudanças**
5. Grande número de pessoas na equipe 6. Grande número de diferentes *stakeholders* na equipe, isto é, apoio de sistemas de informação, usuários, consultores, fornecedores, clientes 7. Tamanho do projeto 8. Grande número de usuário que irão utilizar este sistema 9. Grande número de níveis hierárquicos ocupados por usuários que irão utilizar o sistema, isto é, supervisores	30. O sistema requer um grande número de mudanças em tarefas de usuários 31. O sistema irá conduzir as principais mudanças na organização
Falta de conhecimento geral da equipe	**Recursos insuficientes**
10. Habilidade para trabalhar com objetivo incerto 11. Habilidade para trabalhar com gerenciamento superior 12. Habilidade para trabalhar efetivamente como uma equipe 13. Habilidade para compreender implicações humanas de um novo sistema 14. Habilidade para produzir tarefas efetivamente	32. Para desenvolver e implementar o sistema, o número previsto de pessoas por dia é insuficiente 33. Para desenvolver e implementar o sistema, o orçamento proporcionado é insuficiente
Falta de conhecimento da equipe com a Tarefa	**Falta de clareza nas definições no papel**
15. Conhecimento em profundidade da funcionalidade do departamento usuário 16. Conhecimento global das operações da organização 17. Experiência e perfil em administração geral 18. Conhecimento na área da aplicação específica do sistema 19. Familiaridade com este tipo de aplicação	34. Papel de cada membro da equipe não está claramente definido. 35. Papel de cada pessoa envolvida no projeto não está claramente definido. 36. Comunicações entre os envolvidos no projeto são desagradáveis.
Falta de suporte do usuário	**Complexidade da aplicação**
20. Usuários têm uma opinião negativa sobre o sistema atender suas necessidades 21. Usuários não são entusiasmados com o projeto 22. Usuários não são uma parte integral da equipe de desenvolvimento 23. Usuários não estão disponíveis para responder as questões 24. Usuários não estão prontos para aceitar as mudanças que o sistema irá implicar 25. Usuários respondem morosamente aos pedidos da equipe de desenvolvimento 26. Usuários têm atitudes negativas em relação ao uso de computadores em seu trabalho 27. Usuários não participam ativamente na definição de requisitos	37. Excessivas especificações de requisitos 38. Usuários não estão muito familiarizados com o desenvolvimento de tarefas de sistemas 39. Usuários têm pouca experiência com as atividades que serão suportadas pelas futuras aplicações 40. Usuários não estão muito familiarizados com este tipo de aplicação 41. Usuários não estão conscientes da importância de seus papéis no sucesso completo do projeto 42. Usuários não estão familiarizados com o processamento de dados como um trabalho também

Quadro 4. Instrumento de Medidas de Risco
Fonte: BARKI, H.; RIVARD, S.; TALBOT, J. (1993)

Segundo Hall (1998) os riscos podem ser categorizados como:

a) **riscos de projeto de software:** parâmetros operacionais, organizacionais e contratuais do desenvolvimento de software como, limites de recursos, interfaces, relacionamentos com fornecedores ou restrições de contratos;
b) **riscos de processo de software:** problemas técnicos e de *gerência* como: planejamento, definição e contratação de equipe de trabalho, segurança e configuração de gerência; *técnicos*: análise de requisitos, projeto, codificação e testes;
c) **riscos de produto de software:** tem origem nos requisitos de estabilidade do produto, performance, complexidade de codificação e especificação de testes.

5.3 GERÊNCIA DE RISCOS - NORMAS E MODELOS DE MATURIDADE

As normas e modelos de maturidade de PDS estão relacionadas fortemente com o GR, pois, estas referências defendem que quanto mais organizado for o PDS, menor será o risco de falhas no produto (GUSMÃO; MOURA, 2004).

A *International Organization for Standardization* (ISO) 9000-3 define apenas atividades a serem seguidas para a obtenção da qualidade, dando ênfase a parte contratual. Enquanto a ISO 12207 descreve em detalhes os processos e atividades do ciclo de vida do software para que o desenvolvimento o siga. Já ISO/IEC 15504 apresenta um detalhamento ainda maior dos processos envolvidos no PDS e define como podem ser utilizados para que se obtenha o certificado (GUSMÃO; MOURA, 2004).

Segundo o *Software Engineering Institute* (SEI) cada nível de maturidade está associado a um nível de risco que a organização de software está apta a controlar.

Teoricamente, o nível mais alto de maturidade possui o mais baixo risco para o PDS. Isso porque, quando uma organização avança para posições mais altas nos níveis de maturidade, os riscos são reduzidos a níveis aceitáveis (MYERSON, 1996).

5.4 AVALIAÇÃO DE RISCOS

O primeiro passo para avaliar os riscos é identificar as ameaças presentes no ambiente interno e externo ao PS e analisa-los. A análise dos fatores identificados visa definir as probabilidades de ocorrência e gravidade dos riscos. A análise pode utilizar, para alcançar o objetivo, modelos de performance e custo, análise de redes, análise estatística e qualitativa, entre outras. As padronizações recentes, como o PMBOK e o IEEE, recomendam o uso de procedimentos tanto quantitativos quanto qualitativos, enquanto o CMMI (*Capability Maturity Model Integration*) e a ISO deixam a escolha da abordagem à organização (MACHADO, 2002).

No modo quantitativo há a aplicação de métricas de exposição, já no modo qualitativo pode-se utilizar uma tabela de riscos já identificados e categorizados. A probabilidade multiplicada pelo impacto produz a exposição ao risco, como podemos ver na Figura 21. Em PS, para um evento ou ação ser considerado um risco, deve-se ter uma perda associada, uma chance ou alguma escolha, sendo que o risco pode ser modificado por uma ação (MACHADO, 2002).

$$Ex = L_i * V_i$$

Ex = exposição ao risco
L = probabilidade
V = impacto

Figura 21. Exposição ao Risco
Fonte: MACHADO, C.F. (2002)

Para realizar este cálculo é preciso identificar qual a probabilidade do risco acontecer e se acontecer qual impacto ele vai ter sobre o PS. A obtenção desses valores pode ser feita com um questionário, Figura 22, onde especialistas vão atribuir valores a essas variáveis.

Identificação de Riscos na Manutenção de Software

Analise a lista de fatores de risco abaixo de acordo com a escala definida para **probabilidade** (do risco ocorrer) e **impacto** (caso o risco aconteça). Marque com um X a probabilidade e impacto de cada fator de risco na manutenção de software.

Probabilidade:	**Impacto:**
Muito Provável: (> 70%);	**Catastrófico**: custo do risco excede ao custo planejado do projeto (>= 50%);
Provável: (30% a < 70%);	**Crítico**: custo do risco excede ao custo planejado do projeto (< 50% e >= 10%);
Improvável: (< 30%).	**Marginal**: custo do risco excede ao custo planejado do projeto (< 10%)

Riscos da Manutenção de Sistemas

Categoria de Riscos	Fatores de Risco	Probabilidade			Impacto		
		Muito Provável	Provável	Improvável	Catastrófico	Crítico	Marginal
Sistema a ser mantido	Pouca ou nenhuma documentação Fluxo contínuo de mudanças de requisitos (requisitos instáveis). Efeitos colaterais (impacto) das mudanças em outros sistemas.						
	Efeitos colaterais (impacto) das mudanças em outras funcionalidades do sistema. Baixa qualidade do sistema a ser mantido						

Figura 22. Questionário para a Priorização de Riscos
Fonte: WEBSTER, K.P.B.;OLIVEIRA, K.M.;ANQUETIL,N., (2004)

Com o resultado da exposição ao risco é possível criar uma matriz, conforme Figura 23, para priorizar o atendimento aos riscos. Cada organização deve determinar seus próprios critérios para a classificação das combinações de probabilidade e impacto, que podem ser: alto risco (vermelho), risco mediano (amarelo) ou risco baixo (verde). O cálculo é importante para se saber quais riscos deverão ser tratados, e com que importância LEME (2007).

Escore para um risco específico					
Probabilidade (P)	Escore do Risco = P X I				
0,9	0,05	0,09	0,18	0,36	0,72
0,7	0,04	0,07	0,14	0,28	0,56
0,5	0,03	0,05	0,10	0,20	0,40
0,3	0,02	0,03	0,06	0,12	0,24
0,1	0,01	0,01	0,02	0,04	0,08
	0,05	0,10	0,20	0,40	0,80
	Impacto em um objetivo (I)				

Figura 23. Matriz de Probabilidade X Impacto
Fonte: PMI, (2004)

A tarefa de tratamento de riscos implica em esforço e custo, seja de recursos financeiros e/ou temporais. O tratamento consiste da escolha de ações preventivas e ações de tratamento corretivo, caso o risco venha a acontecer. O gerente deverá escolher qual forma de tratamento ele utilizará para cada risco, são elas LEME (2007):

a) pesquisar: deve-se estudar com relação ao risco para adquirir mais informações e poder determinar melhor as características do risco antes de tomar qualquer iniciativa;

b) aceitar: pode ser possível conviver com as conseqüências de um risco, sem ser necessário tomar outras iniciativas, levando em conta suas variáveis de impacto e probabilidade;

c) evitar: envolve a mudança no plano de gerenciamento de projeto para eliminar a ameaça, isolar objetivos do projeto do impacto do risco, ou ser mais flexível no objetivo que está em perigo;

d) transferir: conferir a um terceirizado a responsabilidade sobre o seu gerenciamento, não elimina o risco;

e) mitigar: gerar ações para redução na probabilidade e/ou impacto de um evento de risco para um valor aceitável, são mais eficazes que tentar reparar o dano;

O guia SWEBOK (2004) define a avaliação dos riscos como o processo de identificar os mais críticos. Deve-se avaliar quanto é critica cada variável de risco. O processo de avaliação deve ser cíclico, já que o ambiente de desenvolvimento é dinâmico e podem surgir e desaparecer novos fatores indesejados a todo o momento, além de alterações relevantes nos graus de impacto. Deve-se classificar o quanto é critico um risco conforme impacto que ele implicará sobre o desenvolvimento e assim determinar qual terá maior urgência de tratamento, a classificação pode ser realizada com o auxilio de um especialista e podem ser:

a) **alto:** este risco terá a maior prioridade;
b) **médio:** este risco terá a prioridade média;
c) **baixo:** este risco terá a menor prioridade.

O método de Avaliação de Risco em Software (SRE - *Software Risk Evaluation*) é uma das técnicas utilizadas para a avaliação da probabilidade de impacto (SISTI; JOSEPH, 1994).

5.5 MÉTODO DE AVALIAÇÃO DE RISCO EM SOFTWARE

O método SRE foi desenvolvido pelo SEI para identificar e reduzir riscos nos estágios iniciais do PDS. O conceito do SRE sugere que, embora bons dados sobre riscos em software possam não estar disponíveis para um projeto em particular, os dados disponíveis de outras experiências em projetos similares podem ser adaptados ou estendidos para uso direto ou como parte de um modelo para revisão dos riscos e seus esforços de mitigação (SISTI; JOSEPH, 1994).

O Paradigma em Gerência de Risco e o Questionário Baseado na Taxonomia (TBQ - *Taxonomy Based Questionary*) são métodos desenvolvidos para apoiar o SRE e devem ser usados na identificação dos riscos (VANSCOY, 1992). Esse método combina *check-list* (questionário) com a técnica de votos múltiplos pela equipe de projeto. O *check-list* serve para explicitar e organizar toda a extensão dos riscos de desenvolvimento de software técnicos ou não. As três classes do TBQ estão divididas em elementos que, por sua vez, são caracterizados por atributos, como mostra a Figura 24 (SISTI; JOSEPH, 1994).

O primeiro elemento, como pode ser visto no TQB, da classe Engenharia de Produto são os requisitos. Os riscos encontrados nesse elemento que não identificados e não tratados poderão se transformar em problemas e proporcionar outros para toda Engenharia de Produto além dos outros elementos de outras classes. Por isso deve-se tratar estes riscos ao final da ER e assim minimizar a geração de riscos para o PS (MACHADO, 2002).

Classe	Engenharia de produto	Ambiente de desenvolvimento	Restrições de programa
Elemento	1. Requisitos	1. Processo de desenvolvimento	1. Recursos
Atributo	a. Estabilidade b. Completeza c. Clareza d. Validade e. Viabilidade f. Precedente g. Escala	a. Formalidade b. Adequação c. Controle do processo d. Familiaridade e. Controle do produto	a. Cronograma b. Equipe c. Orçamento d. Facilidades
Elemento	2. Projeto	2. Desenvolvimento do sistema	2. Contrato
Atributo	a. Funcionalidade b. Dificuldade c. Interface d. Desempenho e. Testabilidade f. Restrições de hardware g. Software não desenvolvido	a. Capacidade b. Adequação c. Usabilidade d. Familiaridade e. Confiabilidade f. Suporte ao sistema g. Entregabilidade	a. Tipo de contrato b. Restrições c. Dependências
Elemento	3. Codificação e teste unitário	3. Processo de gerência	3. Interfaces de projeto
Atributo	a. Viabilidade b. Teste unitário c. Codificação e implementação	a. Planejamento b. Organização do projeto c. Experiência gerencial d. Interfaces de projeto	a. Cliente b. Contratos associados c. Subcontratados d. Principal contratador e. Gerente corporativo f. Vendedores g. Políticos
Elemento	4. Teste e integração	4. Métodos gerenciais	
Atributo	a. Ambiente b. Produto c. Sistema	a. Monitoramento b. Gerência pessoal c. Garantia da qualidade d. Gerência de configuração	
Elemento	5. Especialidade de engenharia	5. Ambiente de trabalho	
Atributo	a. Manutenibilidade b. Confiabilidade c. Segurança d. Proteção e. Fatores humanos f. Especificações	a. Atitude para a qualidade b. Cooperação c. Comunicação d. Moral	

Figura 24. Taxonomia de Risco de Software do SEI
Fonte: SISTI, F. ; JOSEPH, S. (1994)

5.6 RISCOS E REQUISITOS

Os riscos de requisitos estão associados à qualidade da ER e a dificuldade de implementar um software que satisfaça às necessidades do cliente. Diversos são os fatores que contribuem para formação incorreta desta fase como: os requisitos podem ser técnicos difíceis de implementar ou faltar habilidade para negociar requisitos causando eventos negativos (SOARES, 2007).

Os requisitos geram uma fonte de riscos reconhecida pela Gerência de Riscos. Estes riscos podem surgir de processos e técnicas mal realizadas da ER, ou até mesmo pela falta de algumas, gerando perda da qualidade ou até mesmo o cancelamento do projeto. Isso porque alguns modelos de ciclo de vida tratam os riscos do PDS apenas no final, podendo ser tarde demais. Os riscos proporcionados pelos requisitos são (SOARES, 2007):

a) **estabilidade:** alguns PS falham por não haver uma gerência efetiva na evolução dos requisitos visto que ela é inevitável e influência outros fatores do PS;
b) **completude:** requisitos incompletos não conseguem descrever todas as necessidades do cliente e normalmente não indenticam as reais intenções deles, a conseqüência é um escopo que não pode ser alinhado com prazos, orçamentos e recursos;
c) **clareza:** funcionalidade expressada de forma não tão clara passível de interpretação incorreta, a falta de clareza nos requisitos tem sido citada como a terceira maior fonte de riscos encarados por todos os PDS (KENDALL, 2007);
d) **validade:** os requisitos devem refletir as necessidades dos clientes para o software, ou seja, satisfazer as expectativas para ser válido, normalmente este

risco é proveniente do empenho do usuário com o PS, que é citado por Kendall (2007) como a segunda maior fonte de riscos de software;

e) **precedência:** risco do PS propor algo que não foi desenvolvido em softwares existentes ou que são além da experiência da equipe, podem gerar riscos como a não identificação de objetivos inviáveis, dificuldade na implementação;

f) **escala:** se o produto a ser desenvolvido é considerado muito grande e complexo esse risco refere-se a todo o documento de requisitos, ao software formado como um todo, e não a um requisito específico;

g) **viabilidade:** os requisitos precisam ser analisados e negociados junto ao cliente, pois podem ocorrer requisitos inviáveis ao projeto ou até difíceis de implementar.

A ER pode produzir diferentes tipos de risco, por isso deve-se considerar a utilização da Gestão de Riscos ao final dela, pois o produto da ER é a base para o desenvolvimento do PDS. Caso a base do projeto encontra-se incorreta e/ou com riscos provavelmente o software não irá satisfazer todas as necessidades requisitadas pelo cliente corretamente.

6 TRABALHOS CORRELATOS

Os trabalhos abordados a seguir, aplicam técnicas existentes na ES. Essas técnicas devem ser utilizadas no desenvolvimento do software a fim de produzir softwares de maior qualidade que atendam as necessidades do cliente, com o mínimo de falhas possível.

Os diversos trabalhos encontrados relatam diferentes técnicas para a produção dos requisitos. Os diferentes modelos de ciclo de vida do software têm em comum está fase, porém os processos e técnicas podem ser diferentes. O gerente é quem vai determinar quais técnicas serão utilizadas, mas percebe-se que eles têm dificuldades na escolha dos processos que envolvem a produção dos requisitos. Com isso, estratégias estão sendo abordadas para a melhoria dos processos que produzem os requisitos.

Neste sentido, procurou-se pesquisar uma seleção de trabalhos que dos quais trazem consigo técnicas, processos e padrões para uma produção de requisitos com qualidade. Trabalhos que apresentam ferramentas de auxilio ao desenvolvimento do software também foram pesquisados.

6.1 UMA NOVA ABORDAGEM PARA TESTES BASEADOS EM RISCOS NOS REQUISITOS DE SOFTWARE

A pesquisa proposta refere-se a um trabalho de conclusão de curso apresentado como requisito parcial para a obtenção do diploma de Bacharel em Ciência da Computação, realizada na Escola Politécnica de Pernambuco – Universidade de Pernambuco – no ano de 2007 (SOARES, 2007).

O objetivo da pesquisa é realizar uma nova abordagem da técnica *Risk based Testing* (Risco baseado em testes) com o intuito de minimizar as falhas e produzir um software de maior qualidade. Para isso, a técnica será aplicada ao documento de requisitos de um projeto.

O trabalho aborda conceitos de testes e riscos. Os riscos estão classificados e técnicas de identificação foram evidenciadas. Com isso, ele relaciona os testes e riscos e forma os testes baseados em riscos apresentando processos e técnicas para a sua realização.

Posteriormente, é desenvolvida uma estratégia de testes baseados em riscos tendo como a única fonte de riscos um documento de requisitos. A seguir, aplicou-se a estratégia a um documento de requisitos desenvolvido para estudo de caso.

Segundo a autora desse trabalho, os resultados obtidos com o desenvolvimento da estratégia e a sua aplicação, corresponderam às expectativas, sendo os testes baseados em riscos uma ferramenta eficaz na identificação e combate aos riscos gerados pelos requisitos.

6.2 UMA ESTRATÉGIA PARA IMPLANTAÇÃO DE UMA GERENCIA DE REQUISITOS VISANDO A MELHORIA DOS PROCESSOS DE SOFTWARE

O presente trabalho é resultado de um artigo de graduação desenvolvido na Universidade Federal de Pernambuco (CARVALHO; TAVARES; CASTRO, 2004).

O objetivo deste trabalho é definir uma estratégia para que se possa adotar os processos de produção e manutenção de requisitos. A estratégia tem o objetivo de estabelecer um entendimento entre o cliente e a organização que irá produzir o software sobre os requisitos que serão desenvolvidos.

O trabalho aborda a importância da qualidade do software, visto que, ela esta intimamente ligada à produção dos requisitos. Em busca dessa qualidade ele apresenta uma estratégia para a produção da ER. Antes de definir a estratégia ele aborda os vários processos claramente definidos, como: a elicitação, a análise e o gerenciamento dos requisitos. Cada processo tem suas técnicas para obterem o êxito. Assim ele é apresentado uma estratégia de aplicação da ER e comparada com o *Capability Maturity Model* que trata-se de uma métrica para medir o grau de maturidade do projeto e controlar o PDS.

Segundo os autores deste trabalho, o resultado obtido com a implantação da estratégia na ER em um projeto como estudo de caso foi satisfatório, pois apesar das dificuldades a qualidade gerada ao software se mostrou compensatória.

6.3 UMA ANÁLISE CRÍTICA DOS DESAFIOS PARA ENGENHARIA DE REQUISITOS EM MANUTENÇÃO DE SOFTWARE

O presente trabalho é resultado de um artigo desenvolvido na Universidade Católica do Rio Grande do Sul publicado no ano de 2004 no Workshop em Engenharia de Requisitos (ESPINDOLA; MAJDENBAUM; AUDY, 2004).

O objetivo deste trabalho é proporcionar uma análise das principais dificuldades encontradas para realizar a reengenharia de softwares legados. O trabalho identifica dificuldades do de realizar manutenções em softwares sobre a ótica da ER. Estes softwares ainda encontram-se em funcionamento em muitas organizações, pois elas necessitam deles. Essa necessidade faz com que essas organizações procurem melhorar e integrar esses sistemas a conceitos mais atuais.

Os processos de ER fornecidos pela ES foram evidenciados, assim pode-se compreender como cada processo é realizado, bem como algumas técnicas por eles abordadas. Os processos da ER identificados no trabalho visam à produção de requisitos com qualidade.

Segundo os autores deste trabalho, os resultados obtidos foram satisfatórios, pois identificaram os principais desafios da aplicação da ER na manutenção de softwares legados, falta de documentação, falta dos agentes originais que propuseram os requisitos e os requisitos vinculados ao projeto ao invés do produto. Estes três fatores impossibilitam a produção da ER com qualidade, tornando a manutenção alvo de vários riscos.

6.4 GERÊNCIA DE RISCO EM PROCESSOS DE QUALIDADE DE SOFTWARE: UMA ANÁLISE COMPARATIVA

O presente trabalho é resultado de artigo desenvolvido na Universidade Federal de Pernambuco (UFPE), publicado no ano de 2004 no III Simpósio Brasileiro de Qualidade de Software em Brasília (GUSMÃO; MOURA, 2004).

O objetivo deste trabalho é proporcionar uma análise das atividades de Gerência de Risco em alguns modelos de qualidade de software. Visto que o desenvolvimento do software está cercado de incertezas que ocasionam problemas, a utilização da Gerência de Riscos é fundamental para a produção de um software de qualidade.

O presente trabalho aborda diversos conceitos de gerenciamento de riscos, desde Boehm até metodologias mais atuais como o SEI. As atividades em comum são evidenciadas e explicadas. Com isso, são levantadas as atividades de gerenciamento de riscos nos modelos de qualidade como a ISO 9000-3, a ISO 12207 e ISO 15504, *SW-CMM* (*Capability Maturity Model for Software*) e *CMMI*. A seguir, é realizada uma análise para comparar os processos abordados por cada modelo.

Segundo os autores o resultado obtido com a análise definiu que não existe um padrão no gerenciamento de riscos e sim atividades consolidadas como: a identificação de riscos; analisar os riscos; priorizar; controlar os riscos. Sendo que as organizações que a utilizarem, devem ajustar seus processos as atividades do gerenciamento, de acordo com sua realidade.

7 O GERENCIAMENTO DE RISCOS APLICADO A ENGENHARIA DE REQUISITOS PARA EVITAR RISCOS NOS MODELOS DE DESENVOLVIMENTO DE SOFTWARE

O projeto propõe, por meio de um estudo de caso, ratificar a eficácia do controle de qualidade dos requisitos por meio do GR, no sentido de evitar a propagação falhas no PS. Com o apoio da GR será possível visualizar os pontos vulneráveis na formação dos requisitos que tendem a serem grandes problemas posteriormente.

O trabalho irá evidenciar a importância da qualidade de produção dos requisitos durante o ciclo de vida do software, visto a quantidade de pontos falhos que será encontrado em PS desenvolvido sem essa devida conscientização. Serão aplicadas técnicas de GR aos processos de fabricação de requisitos no PS, visando à identificação de falhas e a melhoria continua da ER do projeto. As etapas que vão formar o projeto prático são: o objeto de estudo, a aplicação do gerenciamento de riscos nas fases que geram os requisitos e os resultados obtidos.

7.1 OBJETO DE ESTUDO

O objetivo do trabalho é aplicar conceitos da GR em um PS realizado na disciplina de Engenharia de Software II na Universidade do Extremo Sul Catarinense (UNESC) pela acadêmica Aline Texeira no ano de 2006, que se encontra na integra no Anexo A. Trata-se de um Modulo de Pedidos de Venda para uma empresa do segmento cerâmico, o nome é Sistema de Manutenção de Pedidos de Venda de Revestimentos Cerâmicos (SMPVRV).

O processo de GR terá as fases de identificação e análise dos riscos, além dos planos de ação e controle. As áreas do SMPVRV que serão analisadas são:

a) *Estudo Preliminar com Projeto de Viabilidade*;

b) *Levantamento de Dados*;

c) *Análise de Requisitos*;

d) Modelagem Lógica (Diagramas) e Dicionário de dados;

e) *Planejamento Estratégico da Informação*.

As fases de *Levantamento de Dados*, *Análise de Requisitos* e Modelagem Lógica terão maior ênfase neste trabalho, pela importância que elas têm para o desenvolvimento e pelo fato delas serem o objetivo do estudo. Os riscos que vão prejudicar a formação dos requisitos devem ser tratados, pois certamente implicarão em problemas para o software.

O PS que será utilizado como estudo de caso já foi desenvolvido, então a GR será aplicada não apenas para descobrir riscos, mas também para visualizar falhas que software apresentou depois de pronto.

7.2 PLANEJAMENTO DA GERÊNCIA DOS RISCOS

O GR utilizará para identificar os riscos à técnica de revisão da documentação com o apoio da ferramenta AUDISOFT. Essa ferramenta foi desenvolvida pelo acadêmico Diego Machado Medeiros, como Trabalho de Conclusão de Curso na UNESC no ano de 2008 e trata-se de um software para a auditoria em desenvolvimento sistemas computadorizados. Ela será utilizada para verificar se os processos do documento de requisitos foram realizados, senão irá ajudar a identificar os riscos.

A identificação dos riscos será realizada por partes demonstrando cada trecho do SMPVRV que origina o risco. Os riscos que envolvem os processos como um todo não vão conter trechos do SMPVRV.

Os riscos evidenciados serão abordados em um quadro com os seguintes aspectos: **o risco, a análise do risco, sua categoria, o impacto que ele terá sobre o desenvolvimento e o plano de ação e controle**. Todos os riscos serão mitigados, sendo que, as ações que serão identificas e propostas têm o objetivo de minimizar a ocorrência dos riscos.

Apenas serão abordados nessa fase os riscos considerados de alto impacto para o desenvolvimento, os demais riscos encontrados no projeto podem ser observados no Apêndice A. Atividades de monitoramento e controle serão necessárias para garantir a eficácia das ações de combate aos riscos. Contudo essas atividades serão simuladas, pois o SMPVRV já foi desenvolvido e estas etapas deveriam ser realizadas, paralelamente, com o desenvolvimento do PS. A identificação dos riscos terá inicio com a estrutura do projeto que pode influenciar positivamente ou negativamente o desenvolvimento dele.

7.2.1 Aplicação da Gerência de Riscos no SMPVRV

Nessa fase será aplicado as ações de identificação e combate aos riscos do projeto SMPVRV que visam a melhoria do projeto ainda na fase de geração dos requisitos onde o custo é menor e menos trabalhoso.

7.2.1.1 Estrutura do SMPVRV

O SMPVRV possui as fases básicas de modelagem de desenvolvimento de software. Porém o primeiro risco identificado no SMPVRV foi que ele não adotou nenhum modelo de PDS ou Ciclo de Vida reconhecido cientificamente para realizar o projeto, esse risco é abordado no Quadro 5.

Risco	O PS pode ser desenvolvido incorretamente
Análise do Risco	Apesar do PS ter as fases de produção de software, estas podem ser desenvolvidas incorretamente, visto que, essa produção não percorre os processos e técnicas de um PDS que tenha o reconhecimento cientifico comprovado. Com isso processos podem não serem utilizados no desenvolvimento, além das técnicas que ajudam na formação do PS. Dificultando a produção, gerenciamento e a manutenção do software e desqualificando-o comercialmente.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Adotar um modelo de PDS compatível com software para desenvolvê-lo, Modelo XP foi escolhido pelo software ser pequeno.
Controle	A - Observou-se que os clientes/usuários estão pouco acessíveis e não querem ficar testando protótipos, optou então pela troca do modelo pelo Cascata com realimentação. B - O modelo Cascata com realimentação foi eficaz, minimizado o risco com sucesso.

Quadro 5: Risco de Estrutura

7.2.1.2 Estudo Preliminar e Projeto de Viabilidade

A primeira fase do SMPVRV que será inspecionada é o *Estudo Preliminar com Projeto de Viabilidade*. Os trechos abaixo foram retirados do SMPVRV na integra sendo os quadros de análise de riscos parte constituinte deste trabalho, os riscos são:

a) "Considerando-se que se trata de um estudo acadêmico, não significa que seja a melhor solução encontrada no mercado, porém o projeto é viável tanto economicamente, quanto tecnicamente e legalmente" (TEIXEIRA, 2006, p. 7);

Riscos encontrados:

Risco	Risco de ser inviável tecnicamente.
Análise do Risco	O SMPVRV não se torna viável tecnicamente apenas por ser um estudo acadêmico.
Categoria	Projeto de Software
Impacto	Alto
Planos de Ação	Criar estudo de viabilidade técnica para verificar as técnicas que serão utilizadas na construção.
Controle	A criação de um estudo preliminar da estrutura técnica mostrou-se eficiente e minimizou os riscos de o software obter fracassos na parte técnica.

Quadro 6: Risco de Projeto de Viabilidade 1

Risco	Risco de ser inviável legalmente.
Análise do Risco	O SMPVRV não se torna viável legalmente apenas por ser um estudo acadêmico, pois será desenvolvido para uma organização.
Categoria	Projeto de Software
Impacto	Alto
Planos de Ação	Criar estudo de viabilidade legal para verificar se a construção do software não irá infringir nenhuma lei ou norma da organização.
Controle	O estudo preliminar conseguiu produzir um importante documento de apoio para que o software não sofresse com problemas legais, minimizando este risco.

Quadro 7: Risco de Projeto de Viabilidade 2

Risco	Risco de ser inviável economicamente.
Análise do Risco	O SMPVRV não se torna viável economicamente apenas por ser um estudo acadêmico, pois será desenvolvido para uma organização. Além de já existir um valor para desenvolvimento, implantação e manutenção.
Categoria	Projeto de Software
Impacto	Alto
Planos de Ação	Verificar a viabilidade econômica junto à empresa que vai adquirir o software. Obter a disponibilidade de investimento.
Controle	O estudo minimizou o risco de o software ser economicamente inviável para empresa que vai adquiri-lo.

Quadro 8: Riscos de Projeto de Viabilidade 3

b) "A viabilidade de um software de controle organizacional se torna real por se tratar de um sistema necessário para quaisquer empresas que lidem com fabricação e venda de algum material" (TEIXEIRA, 2006, p. 7).

Riscos encontrados:

Risco	Risco de o SMPVRV não ser implantado na organização
Análise do Risco	O SMPVRV não se torna viável apenas por se tratar de um software de controle organizacional, pois já existem esses tipos de software no mercado. Ele deve apresentar outras vantagens em relação aos similares que certifique sua utilidade para a organização.
Categoria	Projeto de Software
Impacto	Alto
Planos de Ação	Criar um estudo de como o trabalho era realizado antes, sem o software, e depois com o auxilio dele evidenciando os benefícios. Personalizar o software com funcionalidades e/ou aspectos do domínio do cliente.
Controle	O estudo preliminar conseguiu obter funcionalidades e ações específicas da empresa minimizando este risco.

Quadro 9: Risco de Estudo Preliminar

Com os riscos identificados no *Projeto de Viabilidade* percebeu-se que não foi realizada nenhuma forma de obter informações preliminares sobre o software para tornar viável a construção do software. A próxima fase do SMPVRV é o Levantamento de Dados que corresponde ao levantamento de requisitos do software que será revisada no próximo tópico.

7.2.1.3 Levantamento de dados do SMPVRV

A fase de *Levantamento de Dados* do SMPVRV mostrou-se incorreta em vários pontos, identificando que o projetista tem pouca ou nenhuma experiência na área e não se preocupou em buscar ferramentas ou técnicas que o auxiliassem. Os riscos identificados são:

a) "As informações sobre como funcionará esse sistema serão levantadas com auxílio de entrevistas e questionários com um funcionário de uma empresa de revestimentos cerâmicos" (TEIXEIRA, 2006, p. 8);

Riscos encontrados:

Risco	Risco de o software não satisfazer o grupo de usuários.
Análise do Risco	O levantamento apenas utilizou um funcionário para identificar os requisitos, podendo não satisfazer os demais usuários do software, sendo que o questionário e a entrevista não se encontram em anexo ao PS.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Fazer uma reunião com todos os usuários e clientes envolvidos. Eleger quantos representantes for necessário para coleta de informações.
Controle	O plano de ação conseguiu obter vários envolvidos para a formação do escopo de requisitos, diminuindo o risco de ele não satisfazer as necessidades de algum usuário.

Quadro 10: Riscos de Levantamento de Requisitos 1

b) "O sistema será construído com auxílio de um banco de dados e com ele será possível: - Cadastrar, alterar e excluir: Empresas produtoras; TELEFONE, CNPJ.... - Produtos; - Clientes compradores; - Vendedores; - Transportadores; - Países, estados, cidades, bairros e ruas; - Pedidos de Venda. - Realizar consulta das informações contidas do banco" (TEIXEIRA, 2006, p. 8);

Riscos encontrados:

Risco	Risco de clareza dos requisitos
Análise do Risco	Os requisitos funcionais não possuem suas descrições o que pode levar o desenvolvedor a interpretações incorretas.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Abordar cada requisito funcional separadamente, e acrescentar a descrição dele. Identificar lógica de sua criação. Escrever em linguagem natural.
Controle	A ação de descrever e conseguiu esclarecer minimizando o risco.

Quadro 11: Riscos de Levantamento de Requisitos 2

Risco	Risco dos dados armazenados no Banco de Dados serem inconsistentes.
Análise do Risco	Não foram abordados no levantamento requisitos de concorrência na manutenção e inserção de dados no banco, visto que o software oferece acesso simultâneo aos usuários.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Criar requisito não-funcional para tratar a situação de concorrência na consulta e manutenção das informações contidas no Banco.
Controle	Com o tratamento da concorrência este risco foi minimizado.

Quadro 12: Riscos de Levantamento de Requisitos 3

Risco	Risco de clareza no requisito consultas.
Análise do Risco	Deve-se ter um número definido de consultas, pois o cliente pode exigir quantas consultas ele quiser nos termos definidos no levantamento.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Definir uma quantidade de consultas. Definir quais serão as consultas. Abordar elas nos quadros de requisitos respectivamente. Fazer a modelagem utilizando os diagramas UML para melhor compreensão.
Controle	Com as ações propostas as consultas foram desenvolvidas normalmente, minimizando este risco.

Quadro 13: Riscos de Levantamento de Requisitos 4

Com o fim do levantamento dos requisitos pode-se observar que o levantamento não deve ter sido feito com nenhum usuário ou cliente, pois não existem documentos que comprovem o mesmo. Os requisitos levantados são bem genéricos e sem descrições ou detalhamentos podendo todos serem propostos pelo próprio desenvolvedor. Além disso, o

levantamento não identificou nenhuma restrição e nenhum comportamento que software deve ter.

A seguir será revisada a fase de análise dos requisitos levantados, essa fase está bem extensa, pois foi formulado um quadro onde encontram-se a análise, a especificação e os requisitos não-funcionais identificados, posteriormente, pelo projetista.

7.2.1.4 Análise de requisitos do SMPVRV

Nesta fase foi realizado um refinamento dos requisitos levantados e colocado em quadros divididos conforme cada requisito funcional. Possivelmente todas as informações contidas nessa fase devem ter sido geradas, apenas, pelo projetista, pois ele não faz menção nenhuma de ter revisado ou ter analisado os requisitos com o cliente ou usuário.

Os quadros de requisitos, a seguir, foram retirados do SMPVRV na integra, já os quadros de análise de riscos são partes constituintes deste trabalho. Abaixo pode-se observar os riscos encontrados:

a) Requisito Funcional: F1 – Cadastrar País

Descrição: Cadastrar os países que serão utilizados no sistema.

Oculto: Não - Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 1.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		x
NF 1.2 Código do País	O código do país será único, positivo e seqüencial.	Especificação		x
NF 1.3 Cadastro de Dados	Os dados a serem informados no cadastro de país são apenas código e nome do país.	Especificação	X	
NF 1.4 Edição	Após cadastrado, apenas o nome do país poderá ser alterado.	Segurança		x
NF 1.5 Exclusão	Um país poderá ser excluído quando não houver nenhum estado, empresa, cliente, vendedor, transportador relacionado(s) para o mesmo.	Segurança		x

Quadro 14: Cadastrar País - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco do controle de acesso não ser desenvolvido.
Análise do Risco	O requisito controle de acesso é funcional e deve ter um quadro de análise, especificação e requisitos não-funcionais dele.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Abordar o requisito Controle de Acesso como sendo funcional, depois analisá-lo, especifica-lo e identificar seus requisitos não-funcionais, além de fazer a modelagem e constar no dicionário de dados.
Controle	As ações contribuíram para que o Controle de Acesso fosse interpretado corretamente e minimizou-se o risco de ele não ser desenvolvido.

Quadro 15: Risco da Analise de Requisitos 1

b) riscos gerais da análise de requisitos:

Risco	Risco de validade dos requisitos.
Análise do Risco	Os requisitos levantados e analisados não foram validados pelo cliente e podem não serem aceitos posteriormente.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Após montar o escopo de requisitos ele deve ser avaliado e validado pelo cliente ou um representante.
Controle	A - Os clientes não entenderam claramente todos os requisitos, é preciso criar cenários para tirar as duvidas. B - O uso de cenários tornou a compreensão dos requisitos mais fácil sendo eles validados pelo cliente. Assim esse risco foi minimizado.

Quadro 16: Risco da Análise de Requisitos 2

c) riscos gerais de especificação de requisitos

Risco	Risco de completude na Especificação de Requisitos.
Análise do Risco	A especificação está incompleta podendo gerar problemas para o desenvolvimento.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Gerar um formulário padrão para obter a especificação completa com a descrição da função, entradas e origens, saídas e destinos, entidades utilizadas, pré e pós condição e efeitos colaterais se existir.Preencher o formulário em uma entrevista com um representante do usuário adicionar ao quadro de requisitos as especificações descobertas. Gerar diagramas de caso de uso para representar melhor as informações.
Controle	As ações minimizaram o risco e contribuíram para o desenvolvimento correto da Especificação de Requisitos.

Quadro 17: Risco da Análise de Requisitos 3

Risco	Risco de completude da especificação.
Análise do Risco	Especificação incompleta, não se sabe o que acontece com o software se as variáveis receberem valores diferentes dos valores especificados. Esse risco ocorre em todos os campos de preenchimento por parte do usuário no software.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Criar um tratamento para as possíveis entradas que podem ocorrer de forma incorreta. Mostrar mensagem de preenchimento incorreto e limpar o campo para novo preenchimento. Fazer essa correção em todos os campos de que apresentarem esse problema.
Controle	A ação de tratamento das entradas incorretas acabou com o risco do software travar neste momento.

Quadro 18: Riscos da Analise de Requisitos 4

d) riscos de requisitos não-funcionais

Risco	Risco de segurança dos dados armazenados no Banco.
Análise do Risco	Requisitos de segurança para o banco de dados não foram descobertos podendo deixar o banco vulnerável.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Criar um quadro e elicitar requisitos não-funcionais referente ao banco de dados. Criar um controle de acesso ao banco por meio de *login* e senha.
Controle	Com o desenvolvimento do controle de acesso percebeu-se que o administrador do banco continuou usando a senha padrão. Deve-se então tratar a troca de senha no primeiro acesso para não deixar o banco vulnerável. As ações minimizaram o risco de vulnerabilidade das informações do banco.

Quadro 19: Risco da Análise de Requisitos 5

Risco	Risco de o escopo de requisitos estar incompleto.
Análise do Risco	Questões de desempenho, usabilidade, eficiência e portabilidade não foram abordadas.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Realizar nova coleta de requisitos a fim de completar o escopo, com informações sobre desempenho, usabilidade, eficiência e portabilidade. Utilizar um questionário específico.
Controle	A - Será preciso a utilização de cenários e da etnografia. B - Com as técnicas utilizadas foi possível levantar mais requisitos e tornar mais completo o escopo minimizando este risco.

Quadro 20: Risco da Análise de Requisitos 6

Na análise de requisitos podem surgir novos requisitos funcionais e não-funcionais, porém eles devem ter tratamento igual aos requisitos identificados levantamento. Na análise do SMPVRV pode-se observar que requisitos funcionais descobertos nesta fase não obtiveram o tratamento proposto pelo projetista para outros requisitos funcionais. Se outra pessoa fosse codificar este projeto certamente teria muitos problemas para fazê-lo.

A descoberta de novos requisitos que não se encontram no levantamento e a mudança dos existentes deve ser apoiada pela Gerência de Requisitos e suas técnicas. Esses novos requisitos que surgirem devem estar dentro de um período limite para a mudança dos

requisitos, para que não interfiram nas demais fases do projeto, pois eles podem surgir no final do desenvolvimento comprometendo o desenvolvimento. No SMPVRV não foi observado o processo de gerenciamento dos requisitos para descobrir o impacto e identificar a viabilidade das mudanças. Também não houve negociação de requisitos, então subentendesse que nenhum apresentou conflito.

Os riscos e erros identificados nessa fase estão expressos de forma generalizada, pois os quadros de requisitos gerados pelo projetista são bem semelhantes e por isso não foram todos tratados neste momento. Mas eles podem ser observados no Apêndice A junto com os riscos de menor impacto.

Com o fim da fase de Análise de Requisitos foi constatado que a especificação foi realizada, mas está incompleta e os requisitos não-funcionais em sua maioria abordam questões referentes à descrição do requisito funcional em questão. Dando continuidade ao projeto temos a seguir a modelagem dos requisitos que corresponde aos diagramas desenvolvidos.

7.2.1.4 Modelagem dos Requisitos do SMPVRV

A GR será utilizada para identificar e tratar os riscos presentes na modelagem dos requisitos. Os diagramas expressam o comportamento do software em diversas ocasiões, por isso é importante que eles estejam corretos. Os diagramas desenvolvidos no SMPVRV e os riscos descobertos são:

a) diagrama de caso de uso

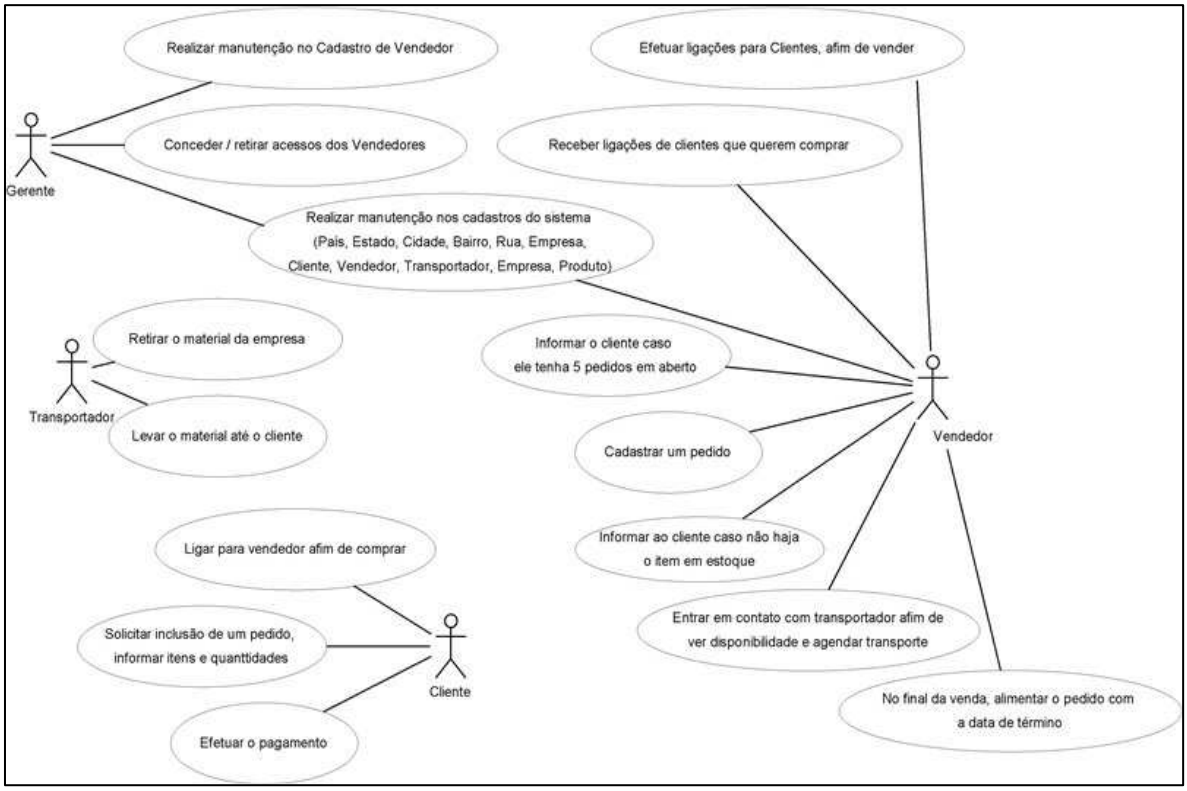


Figura 25: Diagrama de Caso de Uso
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de completude da modelagem dos casos de uso.
Análise do Risco	Falta a narrativa (quadro de descrição) de cada caso de uso do diagrama, pois as informações estão nesses quadros. Dessa forma o diagrama não tem sentido.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Gerar um quadro de descrição para cada caso de uso, com informações como: nome, quem inicia, pré-condição, cenário básico e restrição.
Controle	Com a utilização do quadro de descrição do caso de uso o diagrama ficou completo e compreensível, minimizando este risco.

Quadro 21: Risco no Diagrama de Caso de Uso

b) diagrama de classes

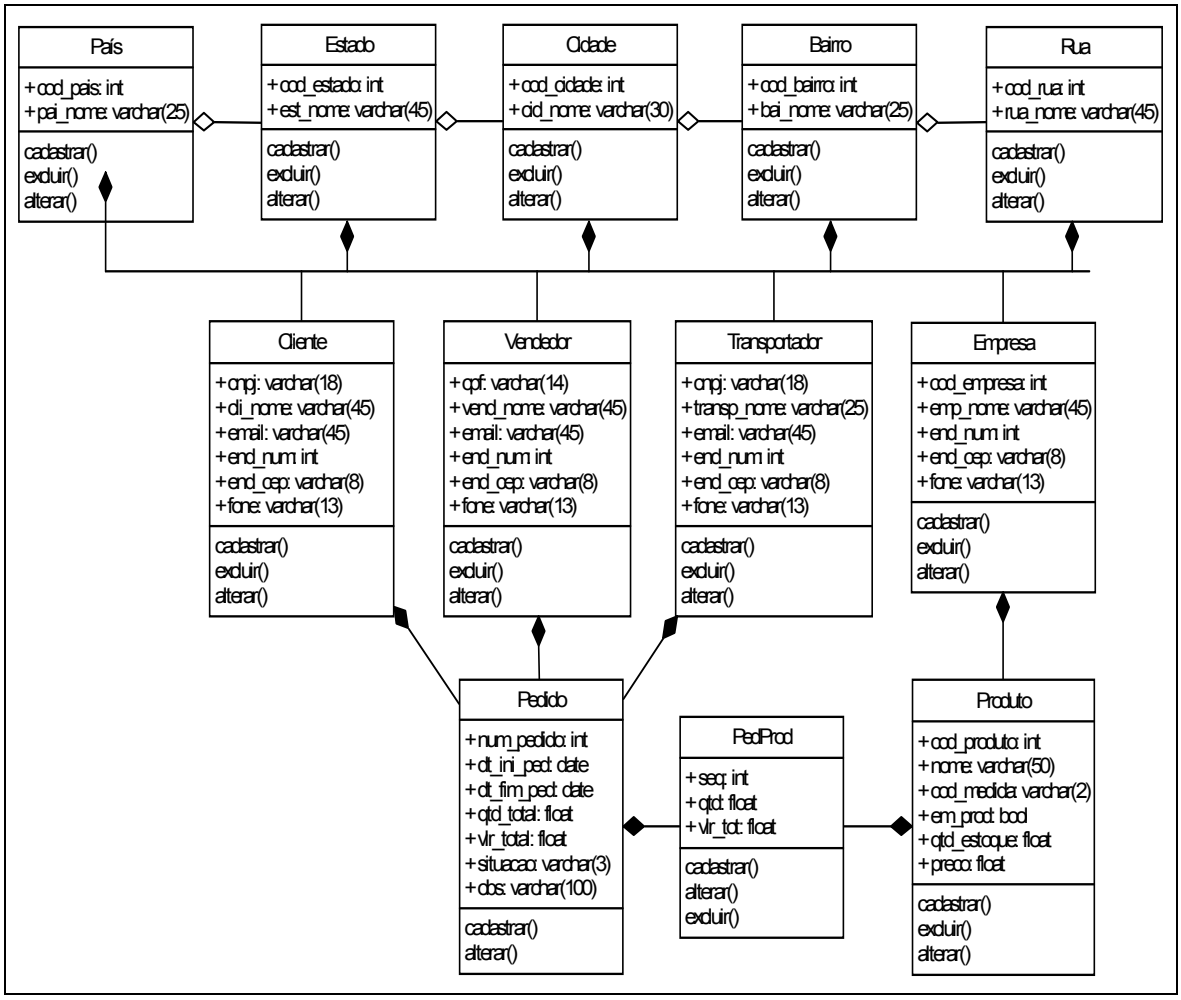


Figura 26: Diagrama de Classes
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco da modelagem das classes estar incorreta.
Análise do Risco	O diagrama de classes está incorreto e confuso prejudicando a interpretação para o desenvolvimento. Há erros nas ligações entre quase todas as classes. O modelo lógico das classes não está normalizado.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Utilizar classes associativas entre País, Estado, Cidade, Bairro, Rua e Empresa e Produto. Normalizar as classes nome, telefone e email. Utilizar classes associativas para telefone e email. Refazer os relacionamentos, seguir modelo de diagrama conforme Apêndice B. Refazer o dicionário de dados tendo como base o novo diagrama de classes.
Controle	A produção de um modelo de diagrama de classes aplicando as ações planejadas minimizou este risco.

Quadro 22: Risco no Diagrama de Classes

c) diagrama de estado – cadastro de cliente

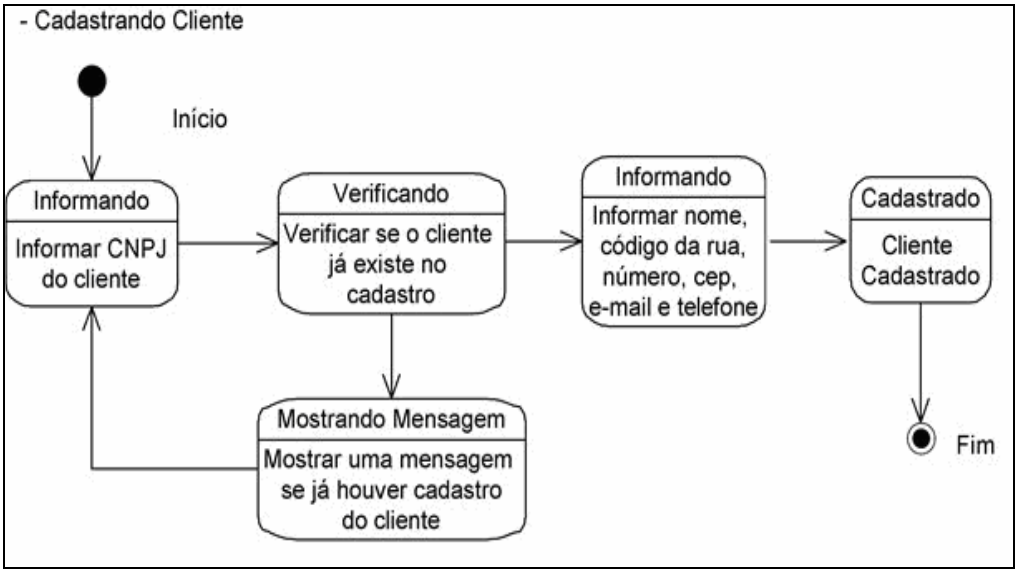


Figura 27: Diagrama de Estado – Cadastro de Cliente
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de completude da modelagem dos estados
Análise do Risco	Os diagramas de estados estão abordando apenas situações simples de cadastro que não seriam necessárias. Porém não existe um diagrama de estado para a *Situação do Pedido* observada no cadastro do pedido que pode passar do estado *cancelado*, *ativo* e *encerrado*. A contagem de pedidos em aberto de cada cliente, que pode chegar apenas a 5 e gera não cadastramento de novos pedidos e uma mensagem ao vendedor avisando-o, seria outro caso para usar este diagrama.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Fazer um diagrama de estado abordando a *Situação do Pedido*. Fazer outro abordando a contagem dos pedidos. Respeitar a estrutura correta do diagrama.
Controle	Com essas ações o risco de completude foi minimizado.

Quadro 23: Risco no Diagrama de Estado – Cadastro de Cliente

d) diagramas de atividade – cadastro de clientes

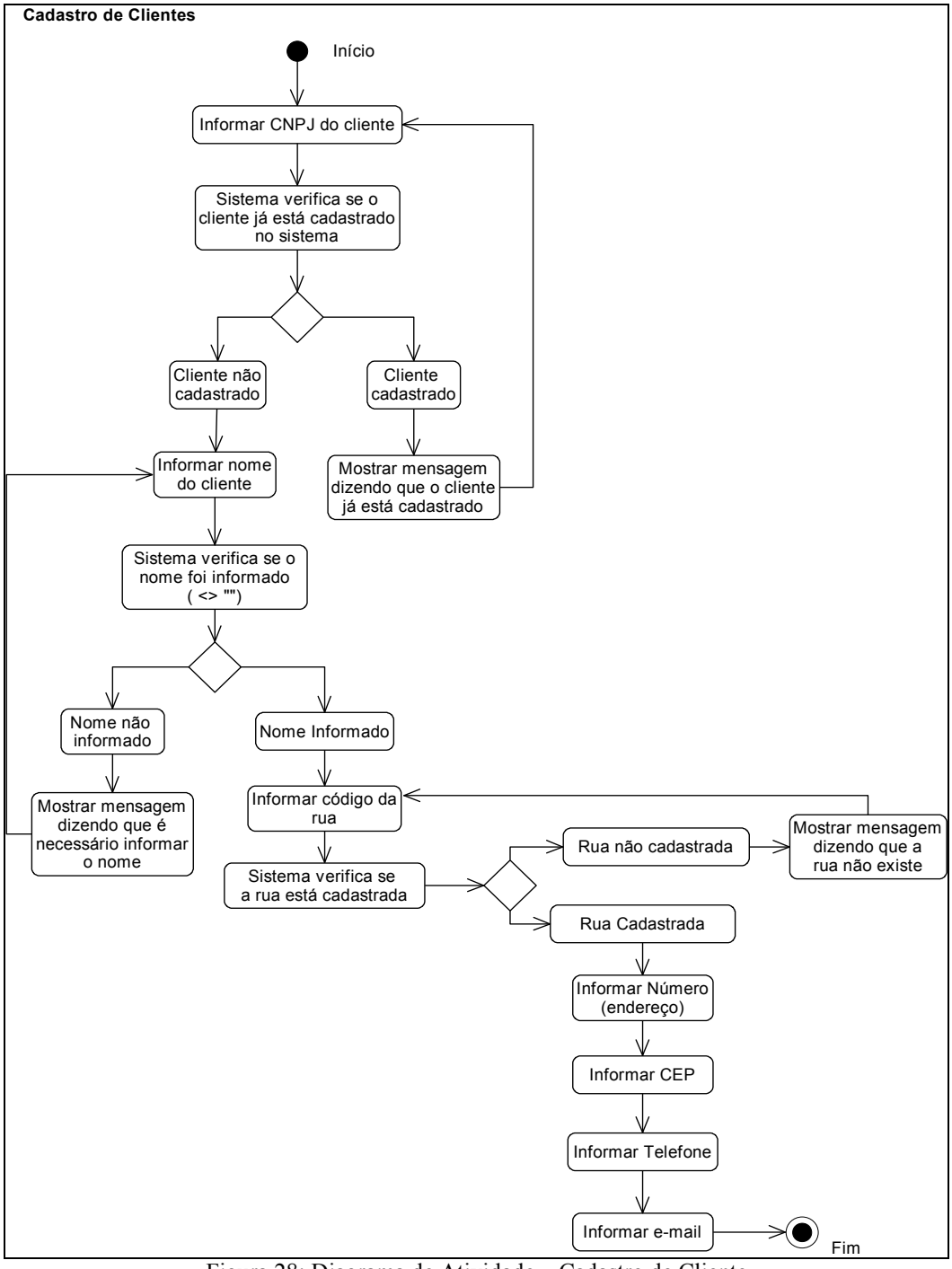


Figura 28: Diagrama de Atividade – Cadastro de Cliente
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de completude na modelagem de atividade.
Análise do Risco	Se o usuário preencher apenas parte do formulário de cadastro pedido pelo cliente, o software não verifica se as variáveis estão nulas. Todas as atividades possíveis no cadastro de cliente devem estar expostas no diagrama.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Verificar ao final do processo de cadastro se as variáveis mais importantes não estão nulas. Se sim mostrar mensagem de cadastro incompleto, senão mostrar mensagem de cadastro com sucesso. Fazer essa correção em todos os Diagramas de Atividade.
Controle	O controle do preenchimento das informações obteve êxito e acabou com este risco.

Quadro 24: Risco no Diagrama de Atividade – Cadastro de Cliente 1

Risco	Risco da modelagem de atividade estar incorreta.
Análise do Risco	O diagrama de atividade de cadastro de cliente deve identificar o país, o estado e a cidade do cliente, conforme os requisitos. As informações rua, país, estado e cidade não podem ser consultadas ou terem novos registros inclusos a partir da janela de cadastro de cliente. Os outros diagramas de atividade de cadastro (Transportador, Empresa e Vendedor) possuem os mesmos problemas.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Incluir nos diagramas de atividade de cadastro os itens a serem informados pelo usuário, país, estado e cidade. As informações rua, país, estado e cidade devem possuir cada uma, um botão que possibilite através de uma janela *pop-up* fazer consulta e se selecionada a informação fechar a janela e auto-completar a caixa de texto com essa informação. Se a informação correta não estiver cadastrada, o usuário deve poder através de um botão, na janela *pop-up* aberta, chamar a janela de cadastro. Integrar as mudanças a análise de requisitos e a modelagem lógica. Essas ações devem ser desenvolvidas também nos diagramas de atividade de Cadastro de Transportes, Empresa e Vendedor.
Controle	Com as ações propostas esse risco foi minimizado.

Quadro 25: Risco no Diagrama de Atividade – Cadastro de Cliente 2

Risco	Risco de desenvolver incorretamente as atividades.
Análise do Risco	A primeira ação deve ser selecionar o que o usuário deseja fazer, incluir novo cliente, alterar, excluir ou consultar o cliente. Este problema esta presente em todos os diagramas de atividade de cadastros.
Categoria	Processo de Software
Impacto	Alto
Planos de Ação	Incluir nos diagramas como primeira ação que o usuário deve fazer a escolha entre incluir, alterar, excluir ou consultar o cliente.
Controle	A correção da ação inicial dos diagramas de atividade de cadastro minimizou este risco.

Quadro 26: Risco no Diagrama de Atividade – Cadastro de Cliente 3

7.2.1.5 Dicionário de Dados do SMPVRV

O dicionário de dados produzido pelo SMPVRV não gerou nenhum risco de impacto alto novo para o desenvolvimento, pois os riscos presentes nesse processo são oriundos da má formação do diagrama de classes do sistema. Porém ele deve ser refeito levando em consideração o novo diagrama de classes que deve ser gerado com as ações propostas para minimizar os riscos.

O dicionário de dados a ser gerado pelo projetista tem varias diferenças em relação a encontrada no SMPVRV. Isso porque classes associativas foram criadas, associações foram modificadas e novas classes surgiram para normalizar os dados.

7.2.1.6 Planejamento Estratégico do SMPVRV

A ultima fase do SMPVRV a ser inspecionada é o *Planejamento Estratégico*. Essa fase é composta pelos requisitos de infra-estrutura administrativa e técnica, os riscos presentes nessa fase estão relacionados a compra de material sem necessidade e falta de outros necessários para a correta implantação do software. Essa não é uma fase prevista na ER mas pode causar transtornos e desperdiço de recursos do cliente senão planejada corretamente. O único risco capaz de causar sérios problemas ao desenvolvimento descoberto nessa fase:

a) infra-estrutura técnica

Riscos encontrados:

Risco	Risco de o desenvolvedor utilizar o banco de dados não compatível com as necessidades do software.
Análise do Risco	O planejamento de infra-estrutura técnica não definiu qual banco de dados será necessário para a implantação do software.
Categoria	Projeto de Software
Impacto	Alto
Planos de Ação	Levantar junto ao cliente e especialistas o banco de dados compatível com o software e as necessidades.Comunicar qual banco de dados será necessário para implantação do software e validação do cliente.
Monitoramento	A ação mostrou-se eficiente.
Controle	Com o esclarecimento o banco pode ser conhecido por todas as partes e aceito, minimizando o risco.

Quadro 27: Risco no Requisito Técnico

7.3 RESULTADOS OBTIDOS

O principal resultado obtido foi a não propagação dos riscos presentes nos requisitos para as demais fases do projeto, com a produção de um documento de requisitos mais confiável. Isso foi possível com a implantação do GR que mostrou-se um processo eficaz no melhoramento do documento de requisitos com a identificação e combate de possíveis problemas ao projeto. Com a aplicação do GR no SMPVRV percebeu-se a quantidade de riscos que o documento de requisitos gerou para o desenvolvimento. Ações de combate propostas poderiam ter evitado vários problemas em situações reais, pois o projeto já foi desenvolvido e muitos desses riscos tornaram-se problemas, afetando diretamente o funcionamento do software.

Além disso, foi obtido um escopo de riscos gerados pelos requisitos na prática, sendo uma grande ferramenta de conhecimento para desenvolvimento de projetos futuros. O SMPVRV demonstra como grande parte das empresas desenvolvedoras de softwares lida com a fabricação da ER, utilizando-se de poucas técnicas, mal estruturadas e mal aplicadas, priorizando o desenvolvimento sem uma documentação consistente.

Com isso, constatou que a pesquisa atingiu o objetivo geral identificando e analisando um método capaz de melhorar a produção da ER e assim evitar que riscos nessa fase se tornem problemas para o projeto.

CONCLUSÃO

Há alguns anos a ER é uma das fases do PDS que mais proporciona problemas ao software, não por ter poucas ferramentas e técnicas, mas porque elas são pouco utilizadas ou mal utilizadas.

O fator tempo foi visto como o grande empecilho para que não haja a produção da ER com qualidade, em conjunto com o pouco comprometimento dos projetistas em realizar a documentação de forma correta e a pouca experiência na área. O projetista não desenvolve uma documentação consistente porque estrutura está fase incorretamente, não aplicando técnicas e ferramentas existentes com intuito de ganhar tempo e chegar logo a fase de programação. Porém ficou claro que o software desenvolvido a partir do documento de requisitos do estudo de caso, apresentou diversos problemas oriundos da documentação ter sido incompleta, inconsistente, pouco clara e modelagem com vários riscos. A seção 2.4 constata estatisticamente que os problemas com requisitos são as maiores causas de problemas do PS, demonstrando que o problema do estudo de caso acontece diariamente.

A seção 8.2.1.3 mostra que o levantamento dos requisitos foi, claramente, mal estruturado utilizando poucas técnicas e identificando poucos requisitos. Além disso, não identificou nenhuma característica do software ou requisitos não-funcionais junto ao cliente, apenas funções. A análise e especificação dos requisitos está incompleta e inconsistente, não conseguindo identificar todas as características de cada requisito e muitas identificadas estão incorretas. A não validação dos requisitos por parte do cliente, que é uma atividade primordial não foi realizada. A modelagem contribui pouco para o entendimento do software, utilizando pouco e de forma incorreta os diagramas da UML. Esses riscos e outros vistos no capitulo 8 e no Apêndice A se transformaram em diversos problemas no software.

Com isso, o projetista levou menos tempo para realizar a ER, porém teria que fazer diversas manutenções no software. Observa-se que a relação custo e tempo seriam maiores se o projetista fizesse a manutenção após o software desenvolvido, ao invés de corrigi-los no documento de requisitos. Por isso foi aplicado a Gerência de Riscos no PS com o objetivo de evidenciar e tratar os riscos gerados na produção da ER. Elucidar os problemas e promover a qualidade da ER ajudando a combater vários outros problemas que se desenvolveriam devido à má formação da ER.

Desta forma, o documento de requisitos abordado no estudo de caso poderia gerar uma base de conhecimento mais consistente e completa, favorecendo o desenvolvimento correto do software por outros agentes que não participaram do projeto e podem se basear na documentação gerada. Assim, todos os objetivos específicos foram alcançados e o objetivo geral foi atingido, conforme o projeto de pesquisa.

Durante o projeto de pesquisa se fez necessário a obtenção de um documento de ER produzido realmente por empresas do segmento, que demonstrariam comercialmente como esse documento é produzido. Mas por se tratar de um documento particular de cada empresa que envolve conhecimentos de um software proprietário, teve-se dificuldades em consegui-lo. Como solução foi utilizado um PS desenvolvido academicamente na UNESC na disciplina de Engenharia de Software II.

Após o entendimento da fundamentação teórica teve inicio a aplicação do GR no PS como estudo de caso. Nesta fase houve dificuldade na aplicação de métricas para analisar a probabilidade dos riscos, pois elas necessitam de conhecimentos de projetos passados que servem de base para o cálculo. Optou-se por utilizar apenas a analise de riscos de forma qualitativa, através do impacto. Teve-se então, dificuldades na identificação de alguns riscos e

na compreensão do impacto deles no PDS. Mas com a ajuda do professor MSc. Paracelso de Oliveira Caldas especialista em ES a identificação pode ser realizada e o impacto estimado.

A presente pesquisa demonstra como se faz necessária a geração de um documento de requisitos de qualidade e técnicas e ferramentas que contribuam para essa concepção devem ser aceitas. Por isso, como trabalho futuro pode-se destacar a geração de uma ferramenta que auxilie na produção do documento de requisitos, oferecendo técnicas e uma base de dados com requisitos já criados. Um outro trabalho futuro seria a aplicação de uma ferramenta de GR que utiliza métricas para demonstrar matematicamente o quão um documento de requisitos pode gerar riscos ao PS.

REFERÊNCIAS

ALMEIDA, Alexandre De; DAROLT, Reginaldo. **Pesquisa e Desenvolvimento em UML**. 2001. 132 f. Tcc (Bacharel) - Curso de Ciência Da Computação, UNISUL – Universidade do Sul de Santa Catarina, Araranguá, 2001.

BARKI, Henri; RIVARD, Suzanne; TALBOT, Jean. **Toward an Assessment of Software Development Risk**. Journal of Management Information Systems, v. 10, n. 2, 1993, p. 203-225.

BEZERRA, Eduardo. **Princípios de analise e projeto de sistemas com UML.** Rio de Janeiro: Campus, 2002.

BOEHM, Barry William. A Spiral Model of Software Development and Enhancement. **IEEE Computer** , v. 21, n. 5, p. 61-72, 1988.

BOEHM, Barry William. **Software Risk Management**. IEEE Computer Society Press: Washington, 1989.

BOEHM, Barry William.; PAPACCIO, Philip N. Understanding and Controlling Software Cost. **IEEE Transactions on Software Engineering** , v. 14, n. 10, p. 1462-1477, 1988.

CAETANO, Cristiano. **Cargos e salários: Quanto ganha o profissional de teste e qualidade de software.** Linha de Código, set. 2004. Disponível em: <http://www.linhadecodigo.com.br/artigos.asp?id_ac=1299 > Acesso em: 02 ago. 2009.

CALDAS JÚNIOR, João; MASIERO, Paulo C**. ERACE-TOOL - Uma Ferramenta Baseada em Cenários para à Engenharia de Requisitos**. Anais do WER98 - Workshop em Engenharia de Requisitos, Maringá-PR, Brasil, Outubro 12, 1998, pp 70-78

CALDAS, Paracelso de Oliveira. **Engenharia de Software I.** 222. Criciúma: Universidade do Extremo Sul Catarinense, 2009.

CARR, Marvin. **Taxonomy Based Risk Identification**. Technical report CMU/SEI- 93-TR-6. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. USA. et al. 1993.

CARVALHO, Ana Elizabete Souza de; TAVARES, Helena Cristina; CASTRO, Jaelson Brelaz. **Uma Estratégia para Implantação de uma Gerencia de Requisitos Visando a Melhoria dos Processos de Software**. 23 f. Artigo (Graduação) - Univesidade Federal de Pernanbuco, 2004.

CASTRO, Marcelo H. **A Volatilidade de Requisitos**. 2009. Disponível em:http://www.ibm.com/developerworks/blogs/page/tlcbr?entry=a_volatilidade_de_requisitos >. Acesso em: 10 jun. 2009.

CHICHINELLI, Micheli; CAZARINI, Edson Walmir. **Requisitos não funcionais e sua importância no processo de desenvolvimento de sistemas de informação**. In: NCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 21, 2001, Salvador, Bahia **Anais...** Bahia: FTC - Faculdade de Tecnologia e Ciências.

CYSNEIROS, Luiz Marcio; LEITE, Julio Cesar Sampaio do Prado. **Definindo Requisitos Não Funcionais**, In: XI SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE. Outubro, 1997, pp. 49-54.

EL EMAM, Khaled; HÖLTJE, Dirk; MADHAVJI, Nazim H. **Causal Analysis of the Requirements Change Process for a Large System**, icsm, pp.214, 13th INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE (ICSM'97), 1997

ESPINDOLA, Rodrigo Santos de; MAJDENBAUM, Azriel; AUDY, Jorge Luiz Nicolas. **Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software**. In: WORKSHOP EM ENGENHARIA DE REQUISITOS, 7., 2004, Tandil. **Anais do WER04.** Argentina: Wer, 2004. v. 9, p. 226 - 238.

FERNANDES, Daniel B**. Especificação de Requisitos**, 2006. Disponível em: < http://imasters.uol.com.br/artigo/4548/des_de_software/especificacao_de_requisitos_por_que_os_projetos_atrasam/>. Acesso em: 23 mar. 2009.

GILB, Tom. **Towards the Engineering of Requirements**. 2003 Disponível em:<www.resultplanning.com> . Acesso em: 23 jun. 2009.

GRUNBACHER, Paul; BRIGGS, Robert O. **Surfacing Tactic Knowledge in Requirements Negotiation: Experiences using EasyWinWin** PROCEEDINGS OF THE 34TH ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, Hawaii, IEEE Computer Society, HICSS 2001

GUSMÃO, Cristine Martins Gomes de; MOURA, Hermano. Perrelli de. **Gerência de Risco em Processos de Qualidade de Software: uma Análise Comparativa**. In: III SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, Brasília, DF. 2004.

GUSTAFSON, David A. **Teoria e problemas de engenharia de software.** Porto Alegre: Bookman, 2003.

HALL, Elaine. **Managing Risk – Methods for Software Systems Developmen**t.1998

HAZAN, Claudia; LEITE, Julio Cesar Sampaio do Prado. **Indicadores para a Gerência de Requisitos**. Anais do WER03 - Workshop em Engenharia de Requisitos, Piracicaba-SP, Brasil, Novembro 27-28, 2003, pp 285-301

HIGUERAG, P.R. **An Introduction to Team Risk Management**, Technical Report. Software Engineering Institute, Carnegie Mellon University. 1994. USA .

IEEE, Institute. **IEEE Standard for Software Maintenance**. New York: Institute of Electrical and Eletronic Engineers. Inc., 1998, 52p.

STANDISH GROUP. **The Chaos Report**, 2009. Disponível em: < http://blogs.msdn.com/b/andredias/archive/2009/07/08/chaos-report-2009-novas-informa-es-velhos-problemas.aspx>. Acesso em: 06 jul. 2010.

JONES, Capers. **Applied Software Measurement: Assuring Productivity and Quality**. New York: McGraw-Hill, 1996b. 618 p.

KEIL, Mark, CULE, Paul E., LYYTINEN, Kalle, SCHMIDT, Roy C. **A Framework for identifying software project Risks**, COMUNICATIONS OF THE ACM, November 1998.

KENDALL, Richard P. **A Proposed Taxonomy for Software Development Risks for High-Performance Computing (HPC) Scientific/Engineering Applications**. Pittburgh, PA: Software Engineering Institute, Carnegie-Mellon University. USA, 2007, et al. 38p.

KOTONYA, Gerald; SOMMERVILLE Ian. **Requirements Engineering: Processes and Techniques**.New York: John Wiley & Sons Ltd, 1998, 282p.

LEME, Lafaiete Henrique Rosa. **Uma Estratégia para Apoiar o Gerenciamento de Riscos em um Ambiente Distribuído de Desenvolvimento de Software.** 2007. 108 f. Dissertação (Mestre) - Universidade Estadual de Maringá, Maringá, 2007.

LEWICKI, Roy J.; LITTERER, Joseph August. **Negotiation,** Homewood III, IRWIN,1985, 368p

LOPES, Paulo Sérgio Naddeo Dias. **Uma Taxonomia da Pesquisa na Área de Engenharia de Requisitos**. 2002. 81 f. Dissertação de Mestrado (Mestre Em Ciência Da Computação) - Universidade De São Paulo, São Paulo, 2002.

LUTZ, Robyn R. **Analyzing Software Requirements Errors in Safety-Critical Embedded Systems**. Proceedings of the ACM SIGSOFT SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING, New York, NY, December 1993, pp. 126-133.

MACHADO, Cristina Filipak. **A-Risk: Um método para Identificar e Quantificar Risco de Prazo em Projetos de Desenvolvimento de Software**. 2002. 239p. Dissertação (Mestrado em Informática Aplicada) – Pós-Graduação em Informática Aplicada - PPGIA, Pontíficia Universidade Católica do Paraná PUCPR, Curitiba.

MYERSON, Marian. **Risk management Processes for Software Engineering Models**. Norwwod: Artech House, 1996.

NOGUEIRA, Admilson. **Casos de Uso – Cenários**, 2006. Disponível em:< http://imasters.uol.com.br/artigo/3811/uml/casos_de_uso_cenarios/ >Acesso em: 03 dez 2009.

NUSEIBEH, Bashar; EASTERBROOK, Steve. **Requirements engineering: a roadmap**. 2000. IN PROCEEDINGS OF THE CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM Press, New York, NY, 35-46.

PAULA FILHO, Wilson de Padua. **Engenharia de Software:** Fundamentos, Metodos e Padrões. Rio de Janeiro: Ltc, 2001. 599 p.

PFLEEGER, Shari Lawrence. **Engenharia de Software:** Teoria e Prática. 2. ed. São Paulo: Prentice Hall, 2004. 535 p.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed São Paulo: McGraw-Hill, 2006. 720p.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995. 1056 p.

PROJECT MANAGEMENT INSTITUTE – PMI. ***A Guide to the Project Management Body of Knowledge*** **(PMBOK)**, PMI Publishing Division, 2004. Disponível em: <http://www.pmi.org> Acesso em: 05 de nov. de 2009.

RAMIRES, João Jorge da Costa Vieira. **Negociação de Requisitos no Processo de Desenvolvimento de Software**. 2004. 179 f. Dissertação de Mestrado (Mestre Em Informática) - Faculdade de Ciências da Universidade de Lisboa, Lisboa, 2004.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação.** 3 ed. Rio de Janeiro: Brasport, 2005.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. . **Qualidade de software:** teoria e prática. São Paulo: Prentice Hall, 2001. 303p.

ROCHA, Álvaro. **Dicionário de Dados**. 2004 Disponível em:< http://www2.ufp.pt/~amrocha/as0506/DicionarioDeDados.pdf>Acesso em: 10 jun 2010.

ROCHA, Pascale Correia; BELCHIOR, Arnaldo Dias. **Mapeamento do Gerenciamento de Riscos no PMBOK, CMMI-SW e RUP**, In: VI SIMPÓSIO INTERNACIONAL DE MELHORIA DE PROCESSOS DE SOFTWARE – SIMPROS, 2004, São Paulo, Brasil.

RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário.** Rio de Janeiro: Ed. Campus, 2000. 472 p.

SAYÃO, Miriam; STAA, Arndt Von; LEITE, Julio Cesar Sampaio do Prado. **Qualidade em Requisitos**, 2003. Monografia - Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) – Rio de Janeiro.

SAYÃO, Mirian; LEITE, Julio Cesar Sampaio do Prado. Rastreabilidade de Requisitos. **Revista de Informática Teórica e Aplicada**, v. 13, n. 1, p. 57-86, 2006.

SCOTT, Kendall. **O Processo Unificado Explicado**. Porto Alegre: Bookman, 2003.

SISTI, Frank; JOSEPH, Sujoe. **Software Risk Evaluation Method**, version 1.0. Pittsburgh, Software Engineering Institute - Carnegie Mellon University, 1994. Technical report CMU/SEI-94-TR-19.

SOARES, Andreza Maria Diniz Morais. **Uma Nova Abordagem para Testes Baseados em Riscos nos Requisitos de Software**. 2007. 91 f. Trabalho de Conclusão de Curso (Graduação) - Univesidade Federal de Pernanbuco, Recife, 2007.

SOMERVILLE, Ian. **Engenharia de Software.** 6 ed. São Paulo: Addison Pearson, 2003.

SOMMERVILLE, Ian. **Software Engineering**, 6ed, Harlow, Addison-Wesley, 2001,742p.

SOMMERVILLE, Ian; SAWYER, Peter. **Requirements Engineering** – a good practice guide.New York: John Wiley & Sons Ltd, 1997, 391p.

SUTHERLAND, Jeff. **SCRUM Software Development Process**. 2001. Disponível em: <http://jeffsutherland.com/scrum/index.html>. 2001. Acesso em: 03 dez. de 2009.

SWEBOK. **Guide to the Software Engineering Body of Knowlegment**, 2004. Disponível em: <http://www.swebok.org/htmlformat.html>. Acesso em 02 dez 2009.

TEIXEIRA, Aline. **Sistema de Manutenção de Pedidos de Venda de Revestimentos Cerâmicos**. 2006. 53 f. Projeto (disciplina de Engenharia de Software I), Universidade do Extremo Sul Catarinense, Criciúma, 2006.

TURINE, Marcelo Augusto Santos; MASIERO, Paulo Cesar. **Especificações de Requisitos**: Uma Introdução. 1996. 26 f. Trabalho de Estágio (Graduação) - Instituto de Ciências Matemáticas de São Carlos, São Carlos, 1996. Cap. 1.

VANSCOY Roger V. **Software development risk**: problem or opportunity. Pittsburgh, Software Engineering Institute - Carnegie Mellon University, 1992. Technical report CMU/SEI-92-TR-30.

VASCONCELOS, Daniel Teófilo**. RUP e XP: uma visão geral**, out. 2005. Disponível em: < http://www.linhadecodigo.com.br/artigos.asp?id_ac=826 >. Acesso em: 01 ago. 2009.

VIANA, Leonardo Mello; DESCHAMPS, Alexandro. **XP – Extreme Programming**. 2008. Disponível em: http://www.apicesoft.com/common/articles/Apice Acesso em: 02 dez. 2009.

VILLER, Stephen; SOMMERVILLE, Ian. **Ethnographically informed analysis for software engineers**. INT. J. HUMAN-COMPUTER STUDIES 53, 2000, 169-196.

VILLER, Stephen; SOMMERVILLE, Ian. **Social Analysis in the Requirement Enginnering Process: From Etnography to Method**, 4th IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENT ENGINEERING. Limerick, Ireland, IEEE Computer Society Press, Los Alamitos, 1999, pp. 6-13.
ZAHEDI, Fatemeh. **Quality information Systems**. Denver: Boyd & Frazer Publishing Company, 1995. 450 p.

ZOWGHI, Didar**;** OFFEN Ray. **A Logical Framework for Modeling and Reasoning about the Evolution of Requirements**, PROCEEDINGS OF THE THIRD IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING (RE'97), pp 247-257, Janeiro 1997, Annapolis, USA.

WEBSTER, Kênia Pereira Batista; OLIVEIRA, Káthia Marçal de; ANQUETIL, Nicolas. **Priorização de Riscos para Manutenção de Software**. 2004. Disponível em:< http://www.grise.upm.es/rearviewmirror/conferencias/jiisic04/Papers/22.pdf >Acesso em: 30 jun 2010.

APÊNDICE A – Riscos de Menor Impacto

1 GERENCIAMENTO DE RISCOS APLICADO A ENGENHARIA DE REQUISITOS PARA EVIDENCIAR RISCOS DE MENOR IMPACTO

Os riscos de menor impacto também podem influenciar a qualidade do software. Identificar, analisar e tratar esses riscos é uma tarefa tão necessária quanto tratar dos riscos de maior impacto. Suas conseqüências são menores caso aconteça, porém seu tratamento ajudará a criar um documento de requisitos mais forte e consistente.

A seguir será continuada a análise do SMPVRV para a descoberta dos riscos de impacto médio e baixo. A metodologia a ser seguida é a mesma proposta para a análise dos riscos de impacto alto.

1.1 ESTUDO PRELIMINAR E PROJETO DE VIABILIDADE

A primeira fase a ser analisada seguindo o projeto é o Estudo Preliminar e Projeto de Viabilidade. Os trechos abaixo foram retirados do SMPVRV na integra sendo os quadros de análise de riscos parte constituinte deste trabalho, os riscos são:

c) “Levando em conta que este software trata apenas do módulo de vendas e considerando a utilização de banco de dados, todo um histórico de vendas poderá ser mantido, podendo-se analisar futuramente através dele quais os produtos mais vendidos, por exemplo, e começar a direcionar produtos conforme a preferência do consumidor” (TEIXEIRA, 2006, p. 7);

Riscos encontrados:

Risco	Risco da tecnologia de produção não ser aceita.
Análise do Risco	Não foi abordado alternativas de tecnologia para o desenvolvimento do produto.
Categoria	Projeto de Software
Impacto	Médio
Planos de Ação	Estabelecer três tipos de tecnologia para desenvolvimento do produto e destacar qual oferece melhor custo/beneficio e é de domínio da organização.
Controle	Com essa ação o cliente pode escolher qual a tecnologia ele deseja para seu software, minimizando o risco.

Quadro 1: Risco de Estudo Preliminar

1.2 LEVANTAMENTO DE REQUISITOS

A segunda fase a ser analisada é o Levantamento de Dados. Os trechos a seguir foram retirados do SMPVRV na integra sendo os quadros de análise de riscos parte constituinte deste trabalho, os riscos são:

c) "As informações sobre como funcionará esse sistema serão levantadas com auxílio de entrevistas e questionários com um funcionário de uma empresa de revestimentos cerâmicos" (TEIXEIRA, 2006, p. 8);

Riscos encontrados:

Risco	Risco das técnicas de levantamento não serem desenvolvidas.
Análise do Risco	Não foram anexados ao PS os documentos que comprovam a aplicação das técnicas.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Anexar os relatórios com as informações coletas, conforme cada técnicas utilizadas para o levantamento de requisitos.
Controle	Com os documentos gerados por cada técnica anexado ao PS algumas duvidas existem sobre os requisitos puderam ser solucionadas com a consulta desses materiais, além do risco dessas técnicas não serem aplicadas ser minimizado.

Quadro 2: Risco de Levantamento de Requisitos

Risco	Riscos das técnicas utilizadas para o levantamento não serem suficientes.
Análise do Risco	Foram utilizados apenas o questionário e a entrevista como técnica para o levantamento de requisitos. O uso de apenas duas técnicas pode comprometer o escopo de requisitos.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Realizar uma reunião com os envolvidos, para detectar requisitos de modo informal. Consultar manuais de outros softwares semelhantes. Observar como as atividades são realizadas na empresa no dia-a-dia. Além das técnicas já propostas.
Controle	Com as ações identificadas este risco foi minimizado.

Quadro 3: Risco de Levantamento de Requisitos

d) "O sistema será construído com auxílio de um banco de dados e com ele será possível: - Cadastrar, alterar e excluir: Empresas produtoras; TELEFONE, CNPJ.... - Produtos; - Clientes compradores; - Vendedores; - Transportadores; - Países, estados, cidades, bairros e ruas; - Pedidos de Venda. - Realizar consulta das informações contidas do banco" (TEIXEIRA, 2006, p. 8);

Riscos encontrados:

Risco	Risco de não haver rastreabilidade dos requisitos.
Análise do Risco	Não foram identificados quem ou o quê originou os requisitos levantados, comprometendo a rastreabilidade do requisito
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Identificar a origem dos requisitos e criar uma árvore de rastreabilidade para apoiar a mudança de requisitos se houver.
Controle	A identificação da fonte e criação da árvore minimizou o risco de rastreabilidade.

Quadro 4: Risco de Rastreabilidade

1.3 ANÁLISE DE REQUISITOS

A fase que será inspecionada agora é a Análise de Requisitos. Os trechos a seguir também foram retirados do SMPVRV na integra sendo os quadros de análise de riscos parte constituinte deste trabalho, os riscos são:

a)

Nome	Restrição	Categoria	Desej.	Perman.
NF 1.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 1.2 Código do País	O código do país será único, positivo e seqüencial.	Especificação		
NF 1.3 Cadastro de Dados	Os dados a serem informados no cadastro de país são apenas código e nome do país.	Especificação	x	

Quadro 5: Cadastrar País - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A (2006)

Riscos encontrados:

Risco	Risco de inconsistência dos requisitos.
Análise do Risco	Existe um conflito entre os requisitos 1.2 e 1.3. O primeiro aborda o código do país como seqüencial e o segundo como sendo informado pelo usuário. Esse mesmo erro é observado nos demais quadros de requisitos.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	O código identificador de ser único, seqüencial e informado pelo sistema. O usuário não poderá modificar este campo. A mesma correção deve ser feita nos outros quadros de requisitos que apresentam o mesmo problema.
Controle	A ação erradicou o risco de contradição.

Quadro 6: Risco de Análise de Requisitos

Risco	Risco de a priorização do requisito estar incorreta.
Análise do Risco	O requisito não-funcional 1.3 não é um requisito desejável e sim um permanente. Essa abordagem incorreta pode confundir o desenvolvedor que pode interpretar erradamente como desejável os requisitos e não implementá-los. Esse mesmo erro é observado nos demais quadros de requisitos.
Categoria	Processo de Software
Impacto	Fraco
Planos de Ação	Corrigir a priorização dos requisitos para permanente. A mesma correção deve ser feita nos outros quadros de requisitos que apresentam o mesmo problema.
Controle	A correção minimizou o risco de o desenvolvedor interpretar incorretamente a priorização.

Quadro 7: Risco de Análise de Requisitos

b)

Nome	Restrição	Categoria	Desej.	Perman.
NF 3.2 Código da Cidade	O código da cidade será único, positivo e seqüencial.	Especificação		X
NF 3.3 Cadastro de Dados	Os dados a serem informados no cadastro de cidade são: código da cidade e nome da cidade.	Especificação	x	
NF 3.4 Edição	Após cadastrada, apenas o nome da cidade poderá ser alterado.	Interface		X

Quadro 8: Cadastrar Cidade - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A (2006)

Riscos encontrados:

Risco	Risco de a categoria estar incorreta.
Análise do Risco	O requisito 3.4 é um requisito de categoria segurança. Esse mesmo erro é observado nos demais quadros de requisitos.
Categoria	Processo de Software
Impacto	Fraco
Planos de Ação	Corrigir a categoria dos requisitos, para não confundir o desenvolvedor. A mesma correção deve ser feita nos outros quadros de requisitos que apresentam o mesmo problema.
Controle	Essa correção minimizou o risco de interpretação incorreta da categoria.

Quadro 9: Risco de Análise de Requisitos

c)

Nome	Restrição	Categoria	Desej.	Perman.
NF 3.2 Código da Cidade	O código do bairro será único, positivo e seqüencial.	Especificação		X
NF 3.3 Cadastro de Dados	Os dados a serem informados no cadastro de bairro são: código do bairro e nome do bairro.	Especificação	x	
NF 3.4 Edição	Após cadastrado, apenas o nome do bairro poderá ser alterado.	Interface		X

Quadro 10: Cadastrar País - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de inconsistência do código identificador do requisito não-funcional.
Análise do Risco	O mesmo código identificador do requisito não-funcional é utilizado no cadastro de cidade, bairro e rua. Isso deve provocar uma confusão na implementação, na rastreabilidade e na contagem de requisitos. Mesmo sendo semelhante cada um deve ter seu código.
Categoria	Processo de Software
Impacto	Fraco
Planos de Ação	Cada requisito não-funcional deve ter seu código de identificação único.
Controle	A ação abordada minimizou este risco.

Quadro 11: Risco de Análise de Requisitos

d)

Nome	Restrição	Categoria	Desej.	Perman.
NF 4.2 Código da Empresa	O código da empresa será único, positivo e seqüencial.	Especificação		X
NF 4.3 Cadastro de Dados	Os dados a serem informados no cadastro de empresa são: código da empresa, rua, bairro, cidade, estado, país, número, CEP, nome da empresa e telefone.	Especificação	x	
NF 4.5 Edição	Após cadastrada, o código da empresa não poderá ser alterado.	Especificação		X

Quadro 12: Cadastrar Empresa - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos identificados:

Risco	Risco de completude dos dados.
Análise do Risco	O requisito não-funcional 4.3 deveria incluir o e-mail, inscrição estadual e cnpj.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Incluir as informações fax, e-mail, inscrição estadual e cnpj no cadastro da empresa.
Controle	As novas informações completam a tela de cadastro minimizando este risco.

Quadro 13: Risco de Análise de Requisitos

e)

Nome	Restrição	Categoria	Desej.	Perman.
NF 5.2 Cadastro de Dados	Os dados a serem informados no cadastro de vendedores são: CPF, rua, bairro, cidade, estado, país, número, CEP, nome, telefone e e-mail.	Especificação	x	
NF 5.4 Edição	Após cadastrado, o CPF de um vendedor não poderá ser modificado.	Especificação		X

Quadro 14: Cadastrar Vendedor - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de o CPF ser um número inexistente.
Análise do Risco	Não existe nenhum tratamento para verificar se o CPF do vendedor é válido.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Criar um requisito de segurança para verificar se o número do CPF é um número válido.
Controle	Com essa ação o risco foi minimizado.

Quadro 15: Risco de Análise de Requisitos

f)

Nome	Restrição	Categoria	Desej.	Perman.
NF 6.2 Cadastro de Dados	Os dados a serem informados no cadastro de transportadores são: CNPJ, rua, bairro, cidade, estado, país, número, CEP, nome, telefone e e-mail.	Especificação	x	
NF 6.5 Exclusão	Um transportador poderá ser excluído se ele não tiver transportado ainda nenhum pedido.	Consistência		X

Quadro 16: Cadastrar Transportador - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de completude dos dados.
Análise do Risco	O cadastro deveria incluir a inscrição estadual como informação a ser coletada. O quadro de cadastro do cliente apresenta o mesmo problema.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Inserir a inscrição estadual como informação a ser requerida nos cadastros de transportador e cliente.
Controle	A ação proposta minimizou o risco.

Quadro 17: Risco de Análise de Requisitos

Risco	Risco de o CNPJ ser um número inexistente.
Análise do Risco	Não existe nenhum tratamento para verificar se o CNPJ da empresa, transportadora e cliente são válidos.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Criar um requisito de segurança para verificar se o número do CNPJ é um número válido. Fazer a correção nos quadros de cadastro da empresa, transportadora e cliente.
Controle	Com essa ação o risco foi minimizado.

Quadro 18: Risco de Análise de Requisitos

g) Cadastrar Pedido – Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 9.2 Número do pedido	O número do pedido será único, positivo e seqüencial.	Especificação		X
NF 9.3 Cadastro de Dados	Os dados a presentes no cadastro de pedidos são: número do pedido, vendedor, transportador, cliente, data de inicio do pedido (inclusão), quantidade total do pedido (soma da quantidade dos itens), valor total do pedido em reais, data término do pedido, situação do pedido (CANcelado, ATIvo, ENCerrado) e observação.	Especificação	x	
NF 9.7 Edição	Após cadastrado, o número do pedido não poderá ser alterado	Interface		X
NF 9.10 Quantidade do produto no pedido	A quantidade do produto no pedido não poderá exceder a quantidade do mesmo em estoque.	Especificação		x

Quadro 19: Cadastrar Pedido - Requisitos Não-Funcionais
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco do requisito não ser desenvolvido.
Análise do Risco	O controle de estoque é um requisito funcional que não foi identificado no levantamento nem na análise dos requisitos e nem foi especificado. A grande maioria das empresas não adota mais o uso dessa função por perceber que a venda pode ser condicionada a um prazo de entrega maior, gerando o tempo necessário para a produção do produto E assim não se perde a venda.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Verificar junto ao cliente se há a necessidade de um controle de estoque. Se sim: identificar mais informações sobre o requisito controle de estoque e analisa-lo e especifica-lo. Gerar um quadro de requisitos não-funcionais para ele. Senão: verificar onde foi utilizado esse controle e troca-lo por uma estrutura que vai gerar o prazo de entrega.
Controle	A ação minimizou o risco da função não ser desenvolvida.

Quadro 20: Risco de Análise de Requisitos

1.3 MODELAGEM DOS REQUISITOS

A Gerência de Riscos vai identificar os riscos presentes na modelagem. Os diagramas desenvolvidos no SMPVRV são:

a) diagrama de Caso de Uso

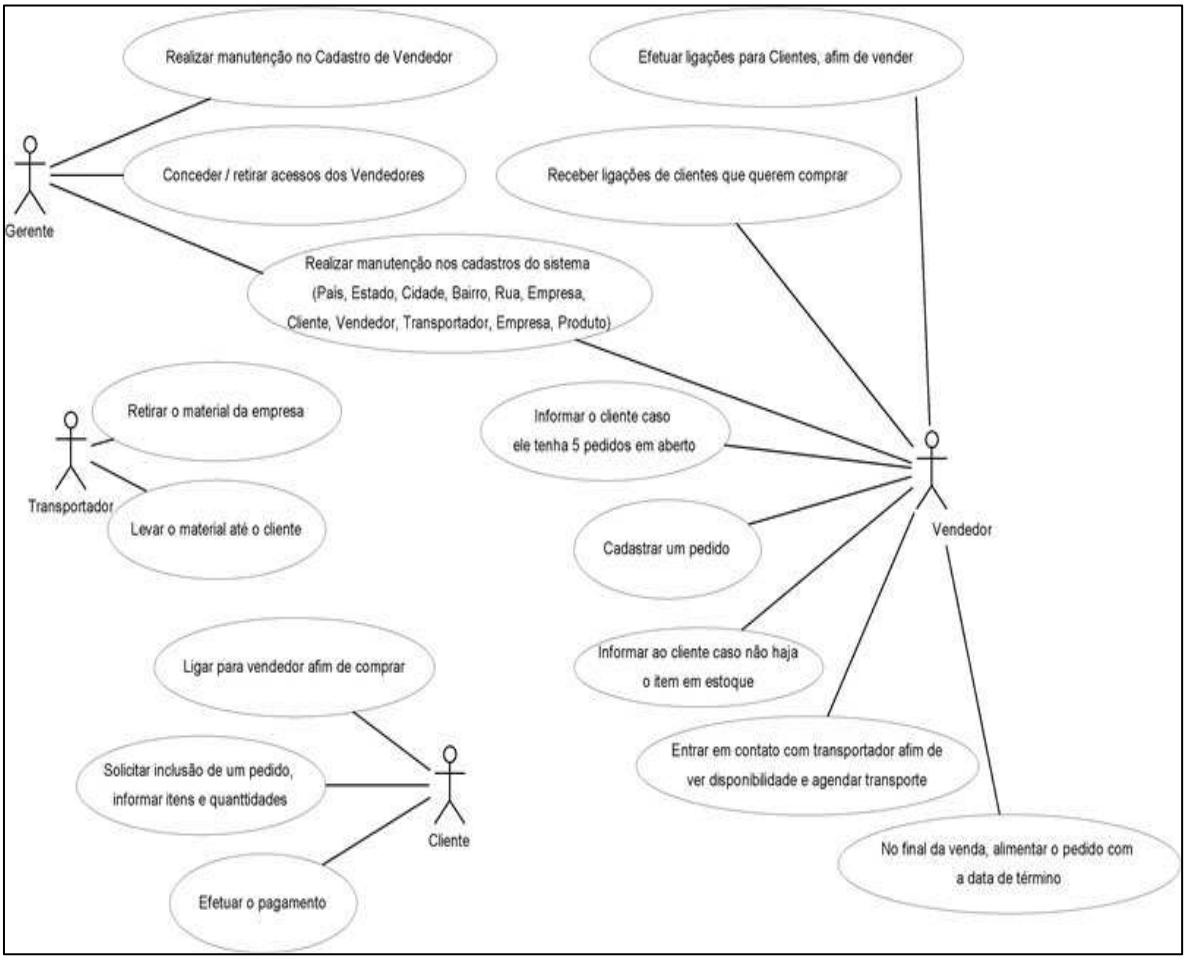


Figura 1: Diagrama de Caso de Uso
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de clareza da modelagem de caso de uso.
Análise do Risco	Atividades que não fazem parte do software foram abordadas no diagrama, tornando-o confuso.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Abordar no diagrama de caso de uso apenas as atividades que fazem parte do escopo de requisitos.
Controle	A ação conseguiu minimizar o risco.

Quadro 21: Risco de Modelagem

b) diagrama de estado – cadastro de cliente

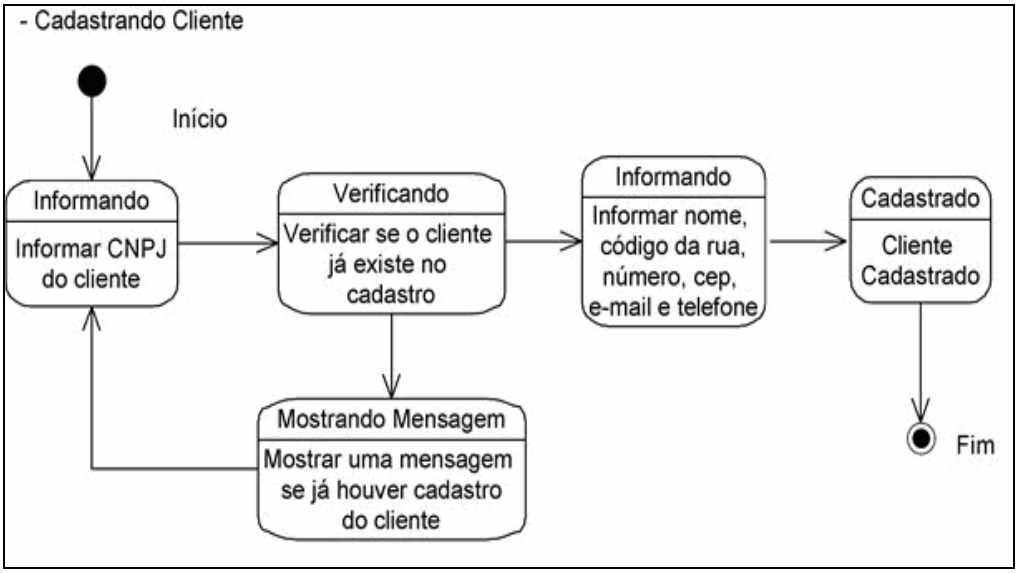


Figura 2: Diagrama de Atividade – Cadastro de Cliente
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de inconsistência na modelagem de estado.
Análise do Risco	Informações que constam no diagrama de estado não são vistas nos quadros de requisitos respectivos. A mensagem é um exemplo. O mesmo problema pode ser vista nos demais diagramas de estado.
Categoria	Processo de Software
Impacto	Baixo
Planos de Ação	Verificar todas as informações que estão nos diagramas de estado e completar o quadro de requisitos e vice-versa. Fazer a verificação em todos os diagramas de estado.
Controle	A ação minimizou o risco.

Quadro 22: Risco de Modelagem

Risco	Risco da modelagem de estados estar incorreta
Análise do Risco	As informações nome, código da rua, numero, cep, e-mail e telefone deveriam ser tratados separadamente cada um em seu quadrante. O mesmo problema pode ser vista nos demais diagramas de estado.
Categoria	Processo de Software
Impacto	Médio
Planos de Ação	Refazer o diagrama de estado respeitando sua fundamentação e abordando as informações que estão aglomeradas cada uma no seu quadrante. Fazer a correção nos diagramas de estado que apresentando o mesmo problema.
Controle	Ação conseguiu dar mais qualidade ao diagrama de estado, minimizando o risco.

Quadro 23: Risco de Modelagem

c) diagrama de atividade – cadastro de pedidos

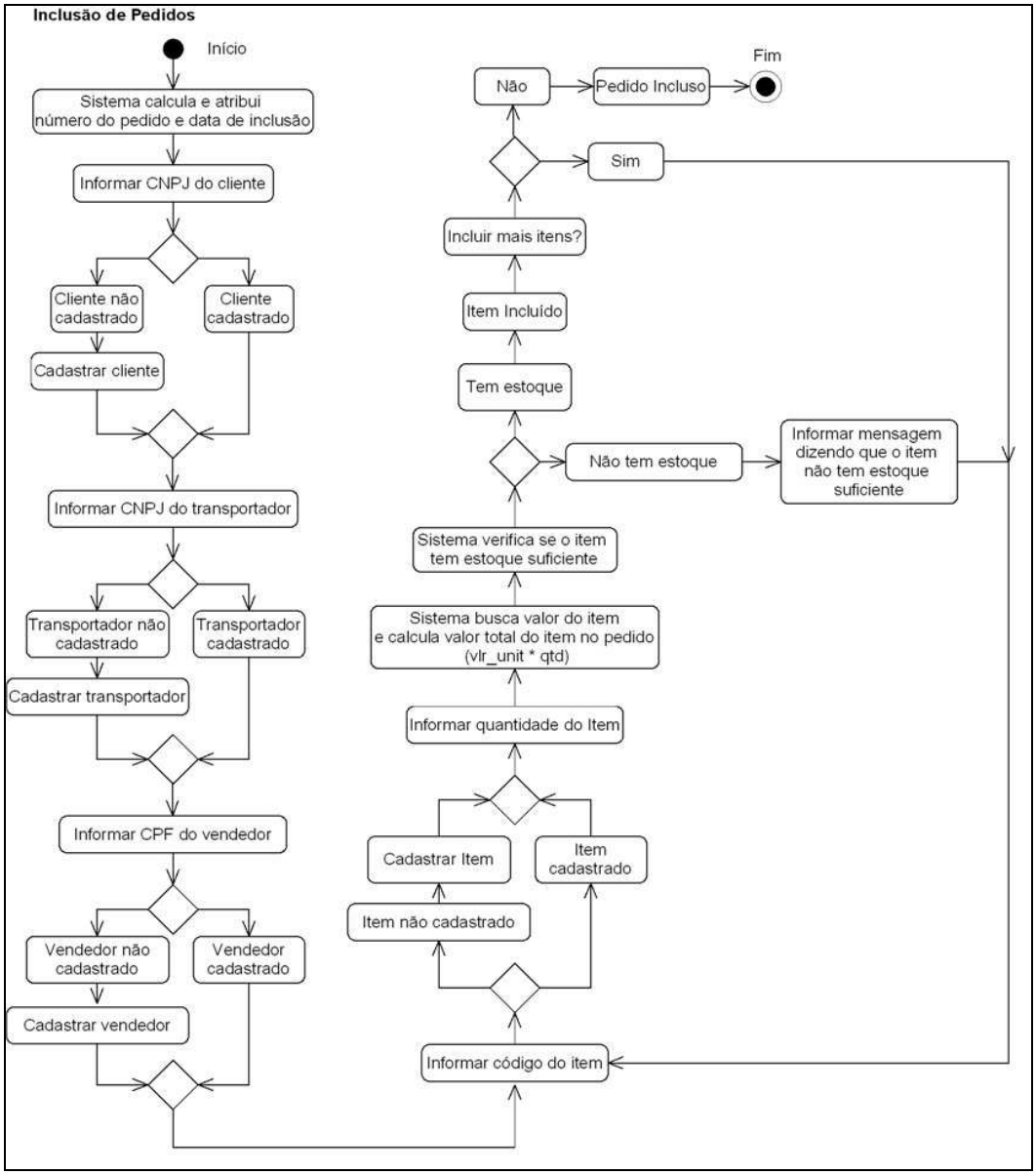


Figura 3: Diagrama de Atividade – Cadastro de Pedido
Fonte: TEIXEIRA, A. (2006)

Riscos encontrados:

Risco	Risco de inconsistência na modelagem de atividades.
Análise do Risco	Informações que constam no diagrama de atividades não são vistas nos quadros de requisitos respectivos. As mensagens, e a situação do pedido são exemplos. Problemas semelhantes podem ser vistos em outros diagramas de atividade.
Categoria	Processo de Software
Impacto	Baixo
Planos de Ação	Verificar todas as informações que estão nos diagramas de atividade e completar o quadro de requisitos e vice-versa. Fazer a verificação em todos os diagramas de atividade.
Controle	A ação minimizou o risco.

Quadro 24: Risco de Modelagem

1.4 PLANEJAMENTO ESTRATÉGICO

A ultima fase do SMPVRV é o Planejamento Estratégico que será analisado. Os trechos abaixo foram retirados do SMPVRV na integra sendo os quadros de análise de riscos parte constituinte deste trabalho, os riscos são:

a) “infra-estrutura administrativa - para a implantação do sistema será necessário um técnico em informática para que instale o software e faça o treinamento para a utilização do mesmo. equipamentos de apoio: 01 Impressora; 04 Mesas para computador; 01 Mesa para impressora; 04 Cadeiras” (TEIXEIRA, 2006, p. 9);

Risco encontrado:

Risco	Risco de o treinamento ser incorreto.
Análise do Risco	O técnico em informática deveria ser treinado, previamente, pela organização desenvolvedora ou ser um técnico em suporte especializado no software para oferecer treinamento aos usuários.
Categoria	Projeto de Software
Impacto	Baixo
Planos de Ação	Utilizar um técnico especializado no software para o treinamento.
Controle	A ação minimizou o risco.

Quadro 25: Risco de Infra-Estrutura

Risco	Risco de utilização incorreta de aparelho.
Análise do Risco	A impressora é um equipamento da área técnica e não de apoio. Um outro equipamento que está na área técnica e é da área de apoio é o Data Show e os discos de *backup*.
Categoria	Projeto de Software
Impacto	Baixo
Planos de Ação	Transferir a impressora para área técnica e os discos de *backup* e o Data Show para a administrativa, pois ele só servirá como apoio para o treinamento. Não é necessário adquiri-lo, apenas loca-lo.
Controle	As ações diminuíram gastos e minimizaram os riscos de uso incorreto.

Quadro 26: Risco de Infra-Estrutura

b) “infra-estrutura técnica - Para que o software seja executado utilizando todos os recursos oferecidos, será necessário 4 microcomputadores com as seguintes configurações mínimas descritas abaixo: Processador Pentium III (700Mhz), ou equivalente, com placa de rede; 64 Mb memória RAM; 40 GB disco rígido; Sistema Operacional mínimo Windows 98; Discos para Backup; 01 Data Show (para treinamento); 01 Switch 24 portas; 01 Modem para acesso ADSL; 01 Bracket 10 U; 01 Patch Panel 24 P; 02 Guias de cabo; 01 Régua de Tomadas (com 6 tomadas); 06 Conectores RJ 45 fêmea; 06 Patch cord 2,5 m; 06 Patch cord 1,5 m; Cabo Multilan 24 AWG 4 pares” (TEIXEIRA, 2006, p. 9).

Riscos encontrados:

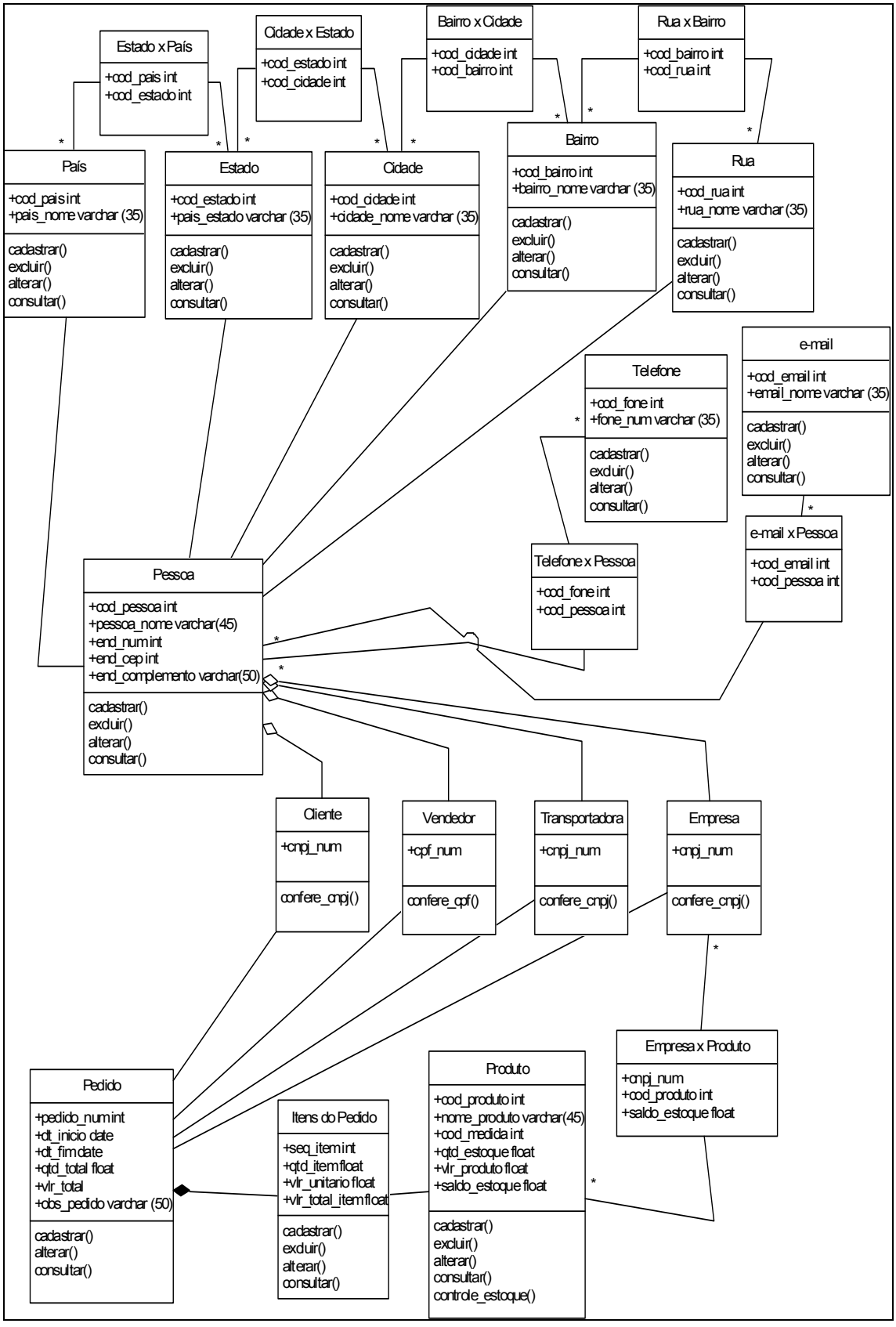
Risco	Risco de Clareza da Infra-estrutura técnica.
Análise do Risco	Não está claro se existirá um computador que servirá os demais, contendo o banco de dados. A configuração dos computadores não especifica a placa de rede e possui pouca memória para um software de janelas com banco de dados
Categoria	Projeto de Software
Impacto	Baixo
Planos de Ação	Identificar se existirá um servidor dedicado, identificar a velocidade da rede para requisitar uma placa compatível e verificar se apenas 64Mb de memória RAM serão suficientes para o uso do software.
Controle	As ações propostas minimizaram este risco.

Quadro 27: Risco de Infra-Estrutura

Risco	Risco de Validade da Infra-estrutura técnica.
Análise do Risco	Alguns equipamentos da infra-estrutura técnica estão em desacordo com o que é necessário para o software.
Categoria	Projeto de Software
Impacto	Médio
Planos de Ação	Um *switch* com 24 portas não seria necessário, pois são apenas 4 computadores e um modem e não existe nenhum plano de expansão; Uma régua com 6 tomadas não servirá para todos equipamentos, rever; Apenas 06 RJs não são suficientes, visto que, são 4 computadores e um switch, rever; A configuração dos computadores deve incluir um monitor, um teclado ABNT, um mouse e um kit multimídia se necessário; Falta cabo de linha telefônica e 2 RJs para conexão do switch com a internet; Seria necessário estabilizadores de corrente elétrica para os computadores e, talvez um *nowbreak* para o servidor;
Controle	As ações previstas minimizaram o gasto com equipamentos incorretos e minimizou o risco.

Quadro 28: Risco de Infra-Estrutura

APÊNDICE B – Modelo de Diagrama de Classes



ANEXO A - SISTEMA DE MANUTENÇÃO DE PEDIDOS DE VENDA DE REVESTIMENTOS CERÂMICOS

ALINE TEIXEIRA

SISTEMA DE MANUTENÇÃO DE PEDIDOS DE VENDA DE REVESTIMENTOS CERÂMICOS

Trabalho destinado à disciplina de Engenharia de Software II, solicitado pelo Prof. Paracelso de O. Caldas.

CRICIÚMA, SETEMBRO DE 2006.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

RESUMO

Este relatório tem como objetivo detalhar o sistema de manutenção de pedidos de venda de uma empresa de revestimentos cerâmicos, visando facilitar o cadastramento da venda e gerenciamento dos dados, bem como apresentar um estudo preliminar da viabilidade do software, avaliação do projeto e cronogramas.

1. INTRODUÇÃO

Toda e qualquer empresa que visa fabricar um produto, também visa vender, que não é a parte mais importante do processo, mas nem tampouco a menos importante.

Vender um produto, propriamente dito, já não é uma coisa muito simples. Numa organização, além da venda, deve haver um controle de tudo que sai, suas quantidades, quem comprou, quem vendeu, quem irá transportar e todas as informações gerenciais necessárias sobre a venda.

E é exatamente isso que propõe o Sistema Manutenção de Pedidos de Venda de uma empresa de revestimentos cerâmicos.

É claro que numa empresa que possui uma certa Estrutura Organizacional, além dos dados de venda, também há controle de estoque, comissão de vendedores, entre outras informações relacionadas a venda, porém não serão tratadas por esse software que cuida exclusivamente do módulo de vendas.

2. ESTUDO PRELIMINAR COM PROJETO DE VIABILIDADE

Partindo da real necessidade de gerenciamento de informações em uma empresa, independente de ser ela pequena, média ou grande porte, fica claro a extrema importância de saber lidar com essas informações, por quais meios elas entrarão no sistema, como serão armazenadas, como serão tratadas e quem terá acesso a elas.

A viabilidade de um software de controle organizacional se torna real por se tratar de um sistema necessário para quaisquer empresas que lidem com fabricação e venda de algum material.

Levando em conta que este software trata apenas do módulo de vendas e considerando a utilização de banco de dados, todo um histórico de vendas poderá ser mantido, podendo-se analisar futuramente através dele quais os produtos mais vendidos, por exemplo, e começar a direcionar produtos conforme a preferência do consumidor.

Considerando-se que se trata de um estudo acadêmico, não significa que seja a melhor solução encontrada no mercado, porém o projeto é viável tanto economicamente, quanto tecnicamente e legalmente.

3. LEVANTAMENTO DE DADOS

As informações sobre como funcionará esse sistema serão levantadas com auxílio de entrevistas e questionários com um funcionário de uma empresa de revestimentos cerâmicos.

O sistema será construído com auxílio de um banco de dados e com ele será possível:

- Cadastrar, alterar e excluir:
    - Empresas produtoras; TELEFONE, CNPJ....
    - Produtos;
    - Clientes compradores;
    - Vendedores;
    - Transportadores;
    - Países, estados, cidades, bairros e ruas;
    - Pedidos de Venda.
- Realizar consulta das informações contidas do banco.

4. PLANEJAMENTO ESTRATÉGICO DA INFORMAÇÃO

4.1. Infra-estrutura administrativa

Para a implantação do sistema será necessário um técnico em informática para que instale o software e faça o treinamento para a utilização do mesmo.

Equipamentos de apoio:

- 01 Impressoras;
- 04 Mesas para computador;
- 01 Mesas para impressora;
- 04 Cadeiras.

4.2. Infra-estrutura técnica

Para que o software seja executado utilizando todos os recursos oferecidos, será necessário 4 microcomputadores com as seguintes configurações mínimas descritas abaixo:

- Processador Pentium III (700Mhz), ou equivalente, com placa de rede;
- 64 Mb memória RAM;
- 40 GB disco rígido;
- Sistema Operacional mínimo Windows 98;

- Discos para Backup;
- 01 Data Show (para treinamento);
- 01 Switch 24 portas
- 01 Modem para acesso ADSL 01 Bracket 10 U
- 01 Patch Panel 24 P
- 02 Guias de cabo
- 01 Régua de Tomadas (com 6 tomadas)
- 06 Conectores RJ 45 fêmea

- 06 Patch cord 2,5 m
- 06 Patch cord 1,5 m
- Cabo Multilan 24 AWG 4 pares

5. ANALISE DE REQUISITOS

5.1. Requisitos Funcionais e Requisitos Não-Funcionais

Requisito Funcional: F1 – Cadastrar País
Descrição: Cadastrar os países que serão utilizados no sistema.

Oculto: Não.

- Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 1.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 1.2 Código do País	O código do país será único, positivo e seqüencial.	Especificação		X
NF 1.3 Cadastro de Dados	Os dados a serem informados no cadastro de país são apenas código e nome do país.	Especificação	x	
NF 1.4 Edição	Após cadastrado, apenas o nome do país poderá ser alterado.	Segurança		X
NF 1.5 Exclusão	Um país poderá ser excluído quando não houver nenhum estado, empresa, cliente, vendedor, transportador relacionado(s) para o mesmo.	Segurança		X

- Requisito Funcional: F2 – Cadastrar Estado
Descrição: Cadastrar os estados que serão utilizados no sistema.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 2.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 2.2 Código do Estado	O código do estado será único, positivo e seqüencial.	Especificação		X
NF 2.3 Cadastro de Dados	Os dados a serem informados no cadastro de estado são: código do estado e nome do estado.	Especificação	x	
NF 2.4 Edição	Após cadastrado, apenas o nome do estado poderá ser alterado.	Segurança		X
NF 2.5 Exclusão	Um estado poderá ser excluído quando não houver cidade, empresa, cliente, vendedor, transportador relacionado(s) com o mesmo.	Segurança		X

- Requisito Funcional: F3 – Cadastrar Cidade
Descrição: Cadastrar as cidades que serão utilizadas no sistema.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 3.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 3.2 Código da Cidade	O código da cidade será único, positivo e seqüencial.	Especificação		X
NF 3.3 Cadastro de Dados	Os dados a serem informados no cadastro de cidade são: código da cidade e nome da cidade.	Especificação	x	
NF 3.4 Edição	Após cadastrada, apenas o nome da cidade poderá ser alterado.	Interface		X

NF 3.5 Exclusão	Uma cidade poderá ser excluída quando não houver nenhum bairro, empresa, cliente, vendedor, transportador relacionado com a mesma	Segurança		X

- Requisito Funcional: F4 – Cadastrar Bairro
Descrição: Cadastrar os bairros que serão utilizadas no sistema.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 3.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 3.2 Código da Cidade	O código do bairro será único, positivo e seqüencial.	Especificação		X
NF 3.3 Cadastro de Dados	Os dados a serem informados no cadastro de bairro são: código do bairro e nome do bairro.	Especificação	x	
NF 3.4 Edição	Após cadastrado, apenas o nome do bairro poderá ser alterado.	Interface		X
NF 3.5 Exclusão	Um bairro poderá ser excluído quando não houver nenhuma rua, empresa, cliente, vendedor, transportador, relacionada com o mesmo	Segurança		X

- Requisito Funcional: F5 – Cadastrar Rua
Descrição: Cadastrar as ruas que serão utilizadas no sistema.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 3.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 3.2 Código da Cidade	O código da rua será único, positivo e seqüencial.	Especificação		X
NF 3.3 Cadastro de Dados	Os dados a serem informados no cadastro de rua são: código da rua e nome.	Especificação	x	
NF 3.4 Edição	Após cadastrada, apenas o nome da rua poderá ser alterado.	Interface		X
NF 3.5 Exclusão	Uma rua poderá ser excluída quando não houver nenhum bairro, cliente, empresa, vendedor ou transportador relacionado com a mesma	Segurança		X

- Requisito Funcional: F6 – Cadastrar Empresa
Descrição: Cadastrar as empresas que serão utilizadas no sistema.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 4.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		x
NF 4.2 Código da Empresa	O código da empresa será único, positivo e seqüencial.	Especificação		x
NF 4.3 Cadastro de Dados	Os dados a serem informados no cadastro de empresa são: código da empresa, rua, bairro, cidade, estado, país, número, CEP, nome da empresa e telefone.	Especificação	x	

NF 4.4 Relação com rua, bairro, cidade, estado, país	Somente empresas cujo código da rua, bairro, cidade, estado, país foram previamente cadastrado poderão ser inclusas no sistema.	Consistência		X
NF 4.5 Edição	Após cadastrada, o código da empresa não poderá ser alterado.	Especificação		x
NF 4.6 Exclusão	Uma empresa poderá ser excluída desde que não haja nenhum item relacionado a ela.	Segurança		x

- Requisito Funcional: F7 – Cadastrar Vendedor
Descrição: Cadastrar os vendedores dos pedidos.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 5.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de gerente de vendas.	Segurança		X
NF 5.2 Cadastro de Dados	Os dados a serem informados no cadastro de vendedores são: CPF, rua, bairro, cidade, estado, país, número, CEP, nome, telefone e e-mail.	Especificação	x	
NF 5.3 Relação com rua, bairro, cidade, estado, país	Somente vendedores cuja rua, bairro, cidade, estado, país foram previamente cadastradas poderão ser inclusos no sistema.	Consistência		X
NF 5.4 Edição	Após cadastrado, o CPF de um vendedor não poderá ser modificado.	Especificação		X
NF 5.5 Exclusão	Um vendedor poderá ser excluído se ele não tiver vendido ainda nenhum pedido.	Segurança		X

- Requisito Funcional: F8 – Cadastrar Transportador
Descrição: Cadastrar os transportadores dos pedidos.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 6.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 6.2 Cadastro de Dados	Os dados a serem informados no cadastro de transportadores são: CNPJ, rua, bairro, cidade, estado, país, número, CEP, nome, telefone e e-mail.	Especificação	x	
NF 6.3 Relação com rua, bairro, cidade, estado, país	Somente transportadores cuja rua, bairro, cidade, estado, país foram previamente cadastrados poderão ser inclusos no sistema.	Consistência		X
NF 6.4 Edição	Após cadastrado, o CNPJ de um vendedor não poderá ser modificado.	Especificação		X
NF 6.5 Exclusão	Um transportador poderá ser excluído se ele não tiver transportado ainda nenhum pedido.	Consistência		X

- Requisito Funcional: F9 – Cadastrar Cliente
Descrição: Cadastrar os clientes que realizam pedidos.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 7.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 7.2 Cadastro de Dados	Os dados a serem informados no cadastro de clientes são: CNPJ, rua, bairro, cidade, estado, país, nome, telefone, e-mail, número e CEP.	Especificação	x	

NF 7.3 Relação com rua, bairro, cidade, estado, país	Somente clientes cuja rua, bairro, cidade, estado, país foram previamente cadastradas poderão ser inclusos no sistema.	Consistência		X
NF 7.4 Edição	Após cadastrado, o CNPJ de um vendedor não poderá ser modificado.	Consistência		X
NF 7.5 Exclusão	Um cliente poderá ser excluído se ele não tiver comprado ainda nenhum pedido.	Consistência		X

- Requisito Funcional: F10 – Cadastrar Produtos
Descrição: Cadastrar os itens fabricados a serem vendidos.

Oculto: Não.

Requisitos Não-Funcionais

Nome	Restrição	Categoria	Desej.	Perman.
NF 8.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 8.2 Código do Item	O código do produto será único, positivo e seqüencial.	Especificação		X
NF 8.3 Cadastro de Dados	Os dados a serem informados no cadastro de produtos são: código do produto, empresa, nome, código de medida (m2, pc, un), se ele está em produção, quantidade em estoque e o preço de venda.	Especificação	x	
NF 8.4 Relação com empresa	Somente produtos cuja empresa foi previamente cadastrada poderão ser inclusos no sistema.	Consistência		X
NF 8.5 Edição	Após cadastrado, o código do produto não poderá ser modificado.	Consistência		X
NF 8.6 Exclusão	Um produto poderá ser excluído se ele não tiver sido vendido ainda em nenhum pedido.	Consistência		X

- Requisito Funcional: F11 – Cadastrar Pedido
Descrição: Cadastrar os pedidos dos clientes.

Oculto: Não.

Requisitos Não-Funcionais

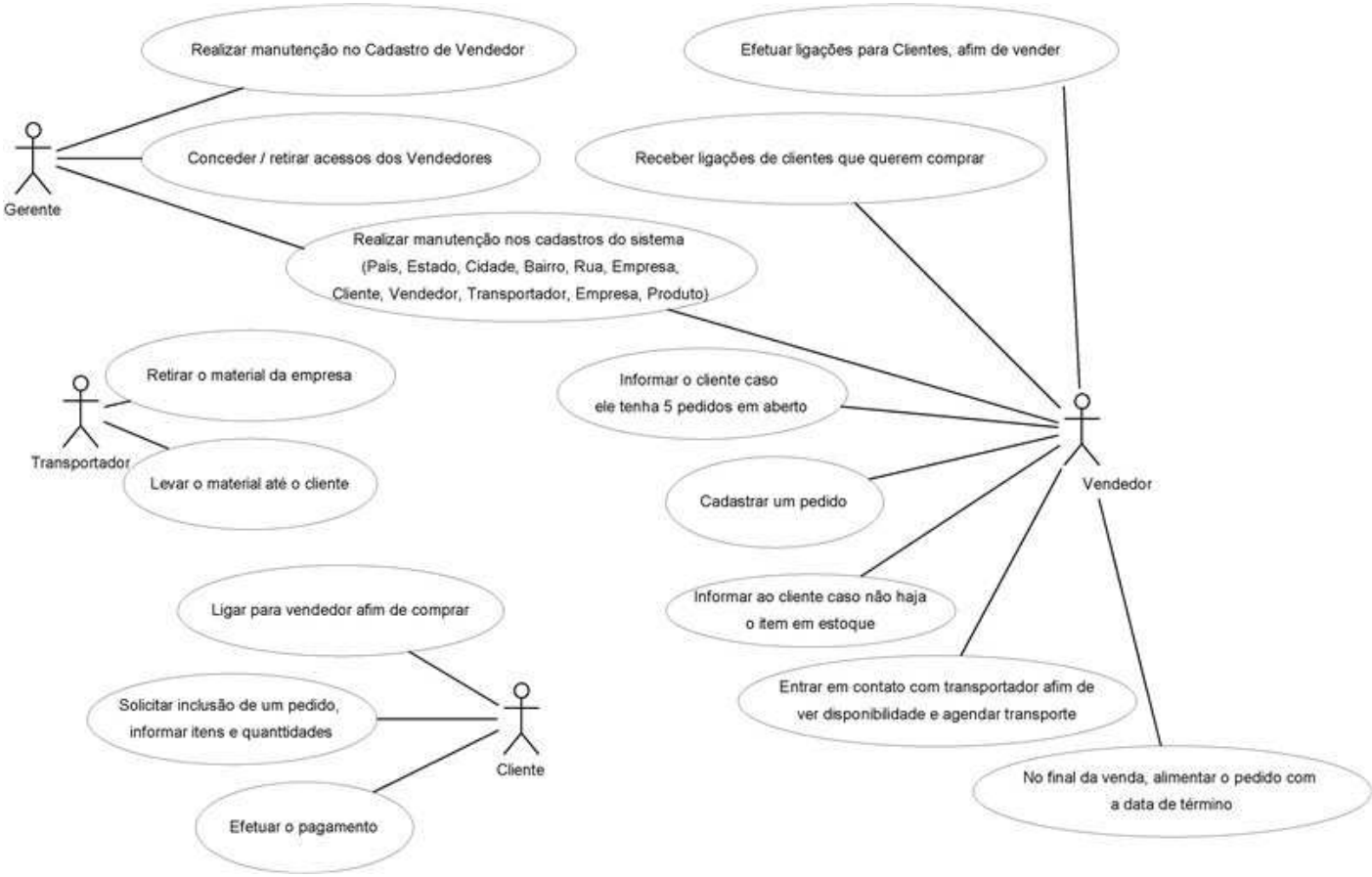
Nome	Restrição	Categoria	Desej.	Perman.
NF 9.1 Controle de Acesso	A função só pode ser acessada por usuário com perfil de vendedor ou gerente de vendas.	Segurança		X
NF 9.2 Número do pedido	O número do pedido será único, positivo e seqüencial.	Especificação		X
NF 9.3 Cadastro de Dados	Os dados presentes no cadastro de pedidos são: número do pedido, vendedor, transportador, cliente, data de inicio do pedido (inclusão), quantidade total do pedido (soma da quantidade dos itens), valor total do pedido em reais, data término do pedido, situação do pedido (CANcelado, ATIvo, ENCerrado) e observação.	Especificação	x	
NF 9.4 Relação com vendedor	Somente pedidos cujo vendedor foi previamente cadastrado poderão ser inclusos no sistema.	Consistência		X
NF 9.5 Relação com transportador	Somente pedidos cujo transportador foi previamente cadastrado poderão ser inclusos no sistema.	Consistência		X
NF 9.6 Relação com cliente	Somente pedidos cujo cliente foi previamente cadastrado poderão ser inclusos no sistema.	Consistência		X
NF 9.7 Edição	Após cadastrado, o número do pedido não poderá ser alterado	Interface		X
NF 9.8 Número de pedidos por cliente	Um cliente não pode ter mais de 5 pedidos em aberto.	Especificação	x	
NF 9.9 Produtos do Pedido	Na inclusão do pedido informar-se-ão os produtos pertencentes ao mesmo, dando origem a uma outra estrutura (Pedido x Produto), onde serão informadas as quantidades de cada produto e	Especificação		X

	o valor total daquele produto no pedido (quantidade * preço de venda).			
NF 9.10 Quantidade do produto no pedido	A quantidade do produto no pedido não poderá exceder a quantidade do mesmo em estoque.	Especificação		X

6. DIAGRAMAS DE CASOS DE USO

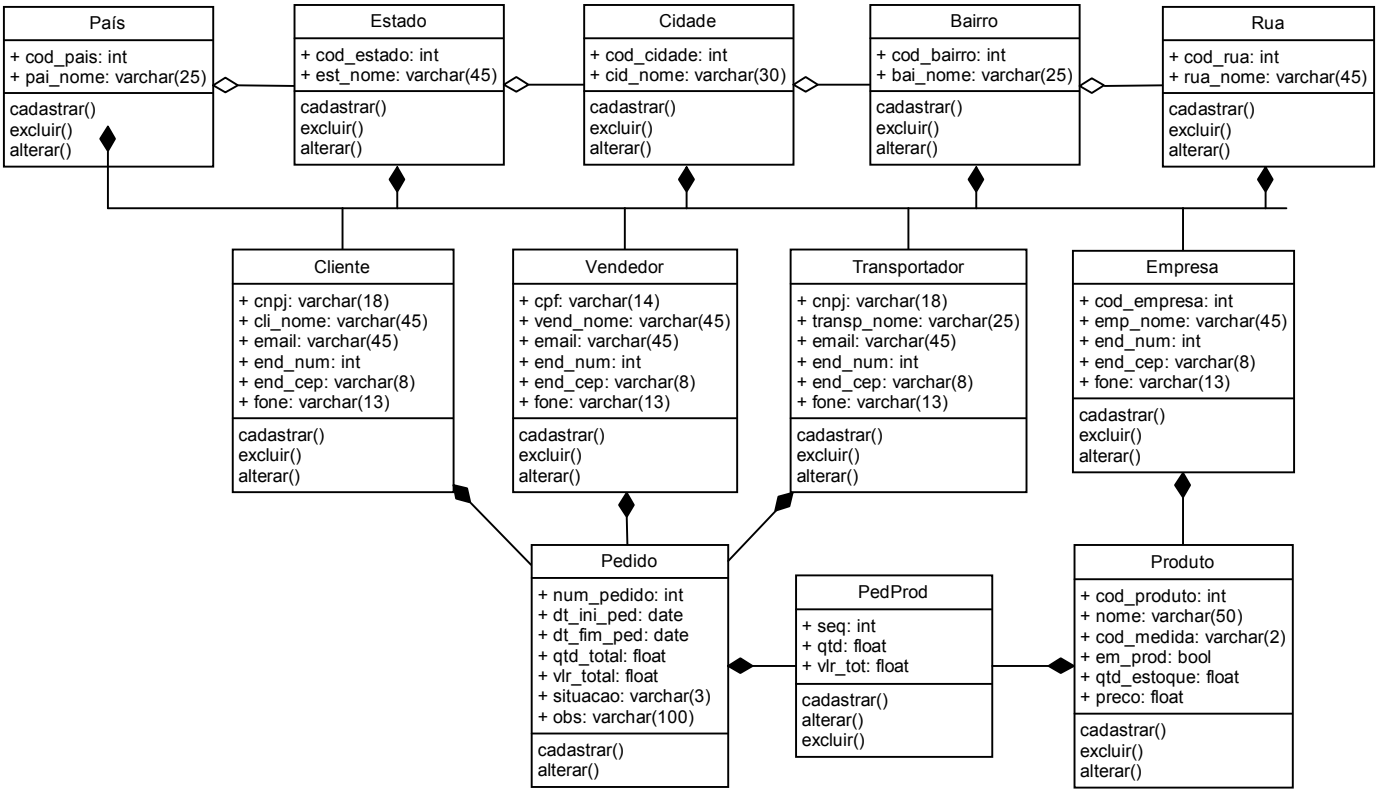
Os diagramas de Casos de Uso (Use Case) representam o comportamento (ações) de um usuário perante ao sistema, sendo que o usuário pode ser uma pessoa ou uma máquina.

É composto por atores ( ) e use cases ( ). Os atores representam as entidades que interagem com o sistema e as use cases mostram a seqüência de ações que o sistema executa.



7. DIAGRAMA DE CLASSES

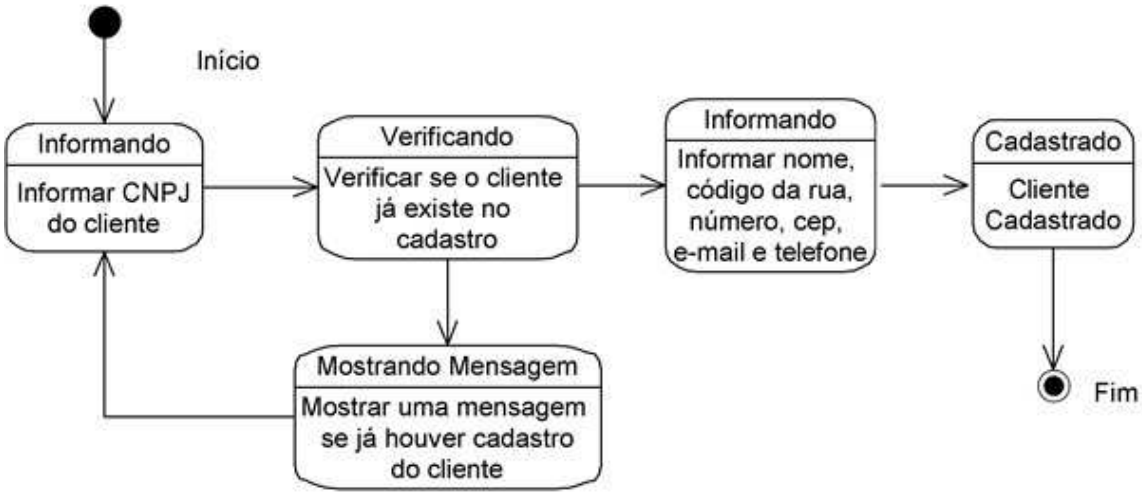
Este diagrama, encontrado geralmente na modelagem de dados de sistemas orientados a objetos, representa as classes do sistema, com seus atributos e métodos, bem como os relacionamentos entre as classes. Os losangos preenchidos representam uma relação de composição, ou seja, uma classe depende da outra e sem ela não faz sentido, deixando de existir (Para existir PedProd é necessário existir Pedidos e Produtos). Já os não preenchidos representam a agregação, ou seja, elas estão relacionadas mas também existem separadamente.



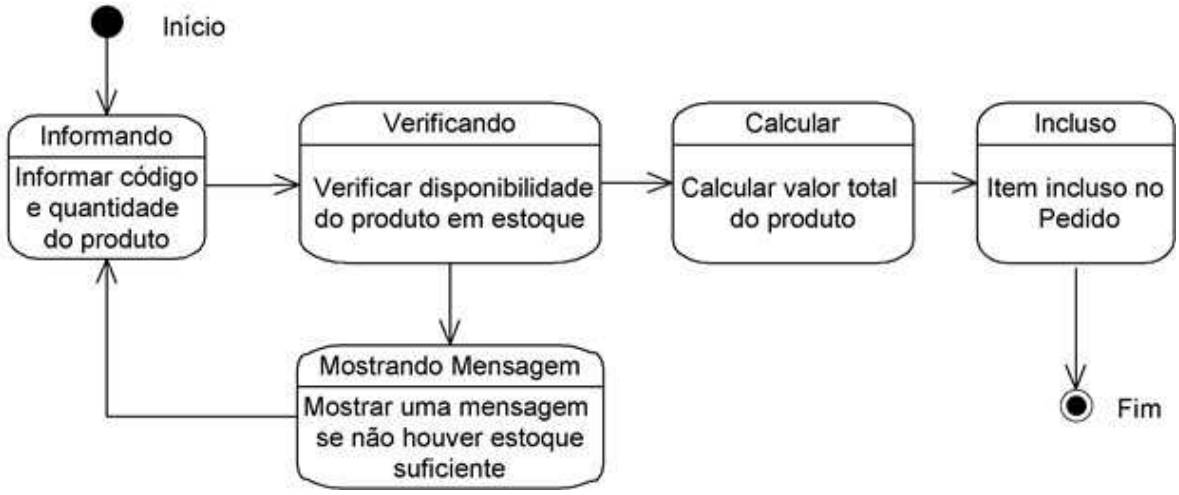
8. DIAGRAMAS DINÂMICOS

8.1. Diagrama de Estados

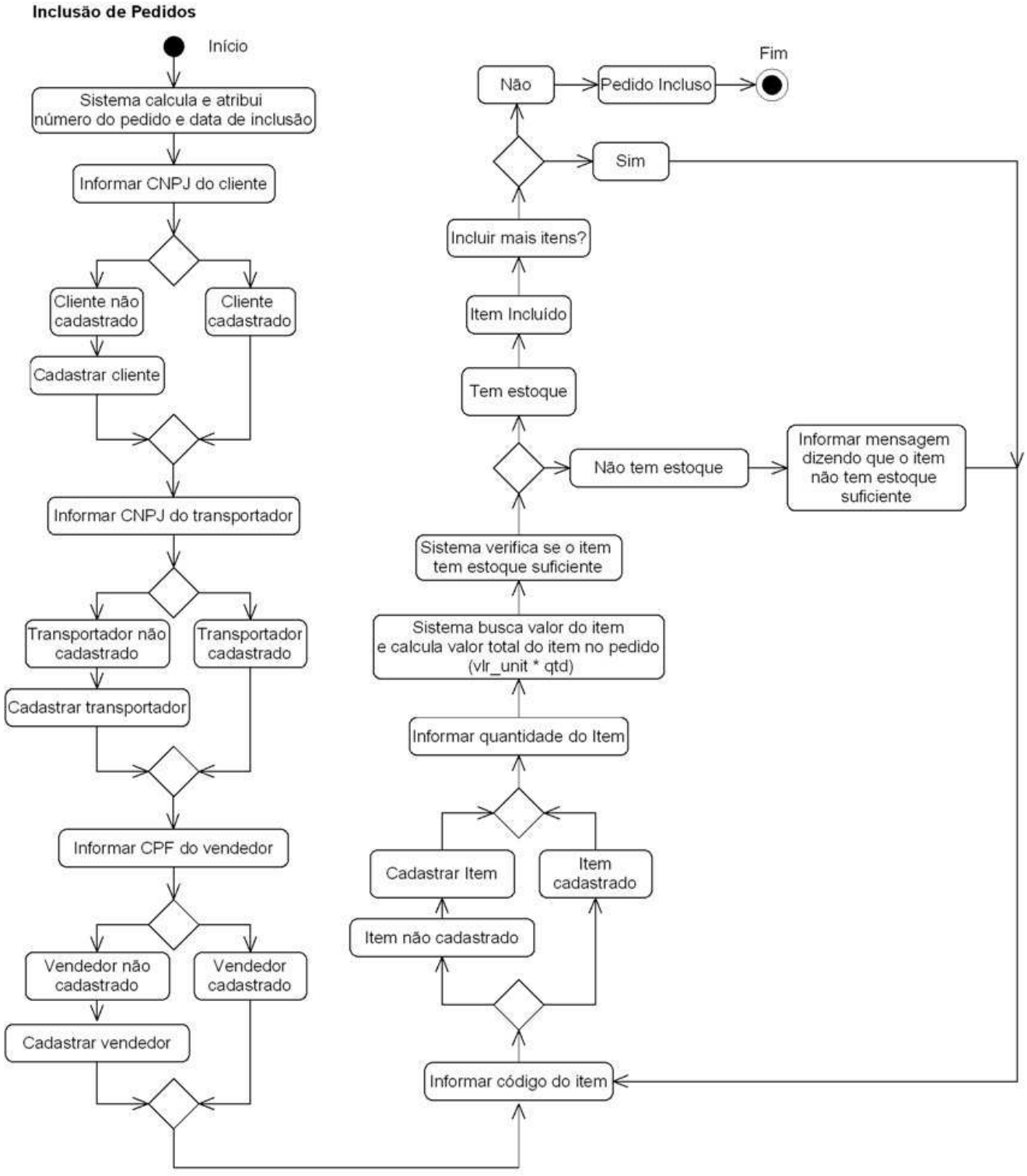
- Cadastrando Cliente



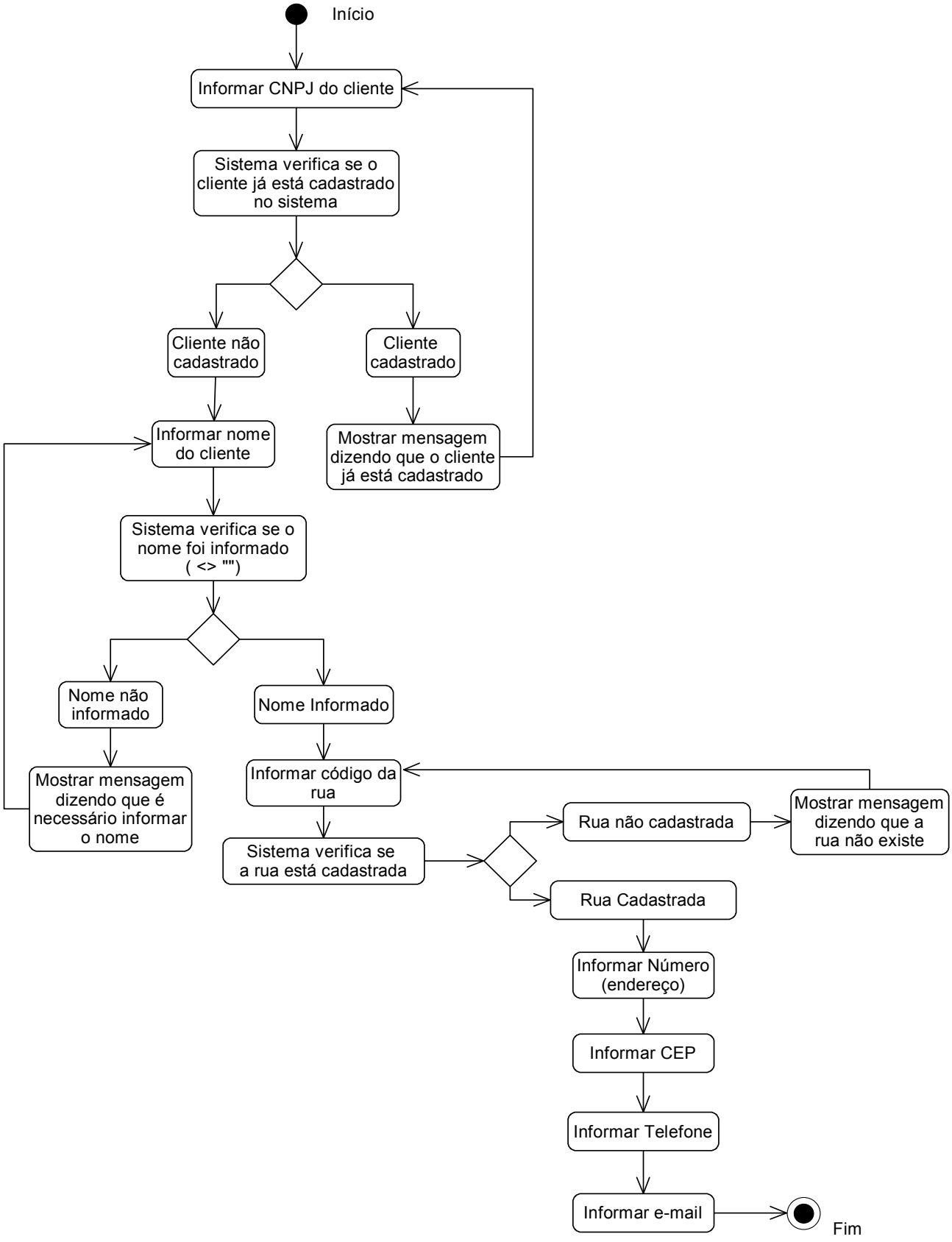
- Incluindo Itens no Pedido



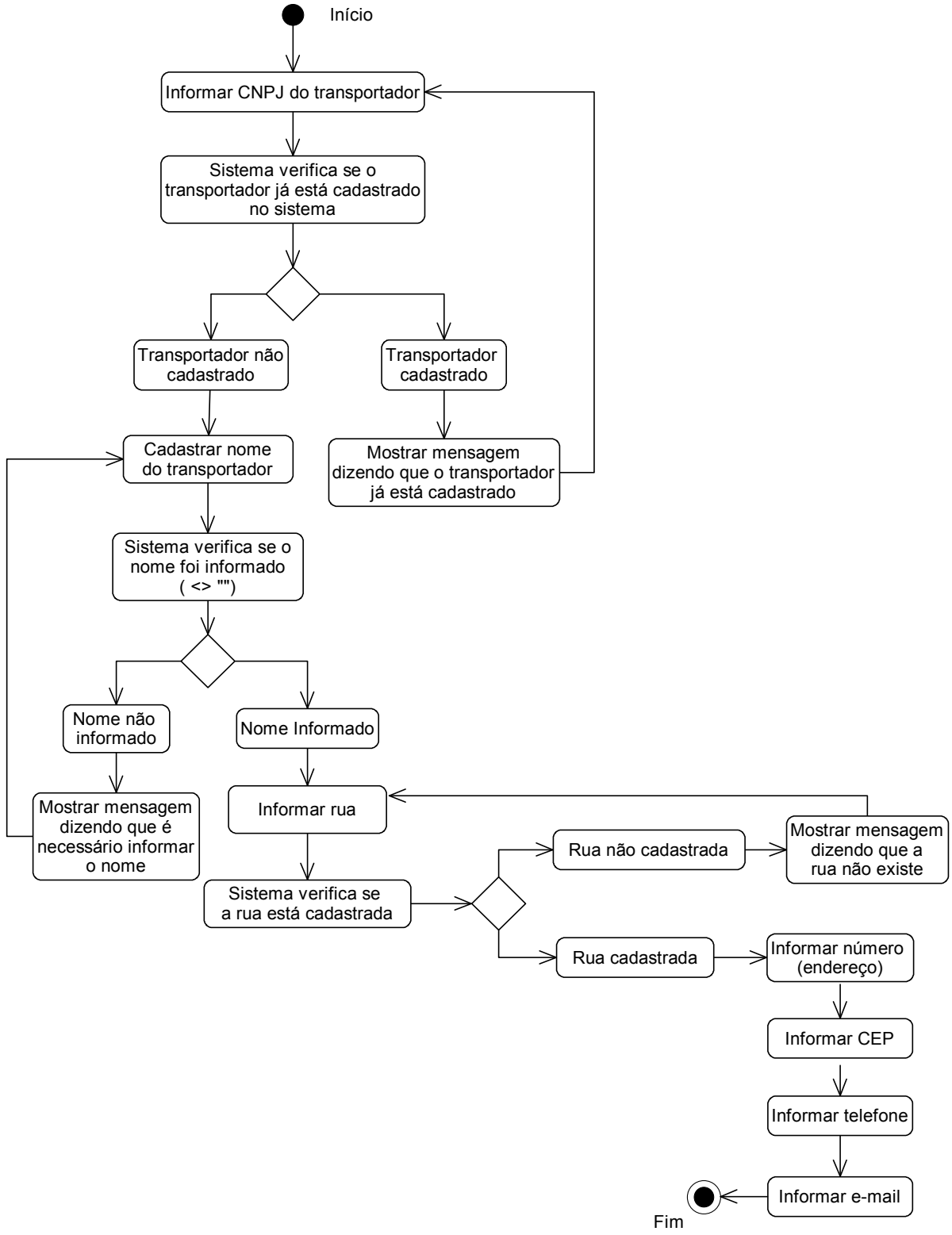
8.2. Diagrama de Atividades



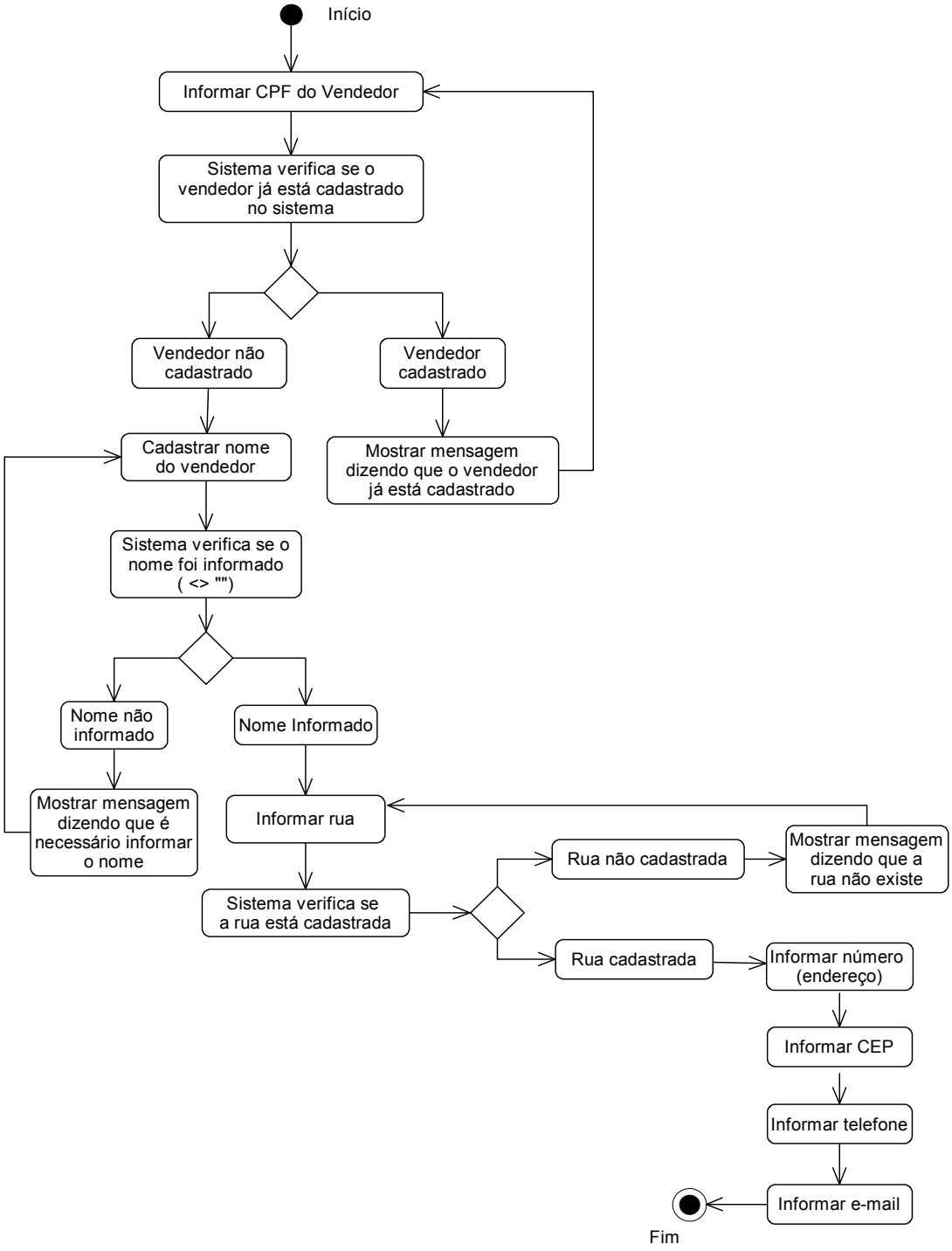
Cadastro de Clientes



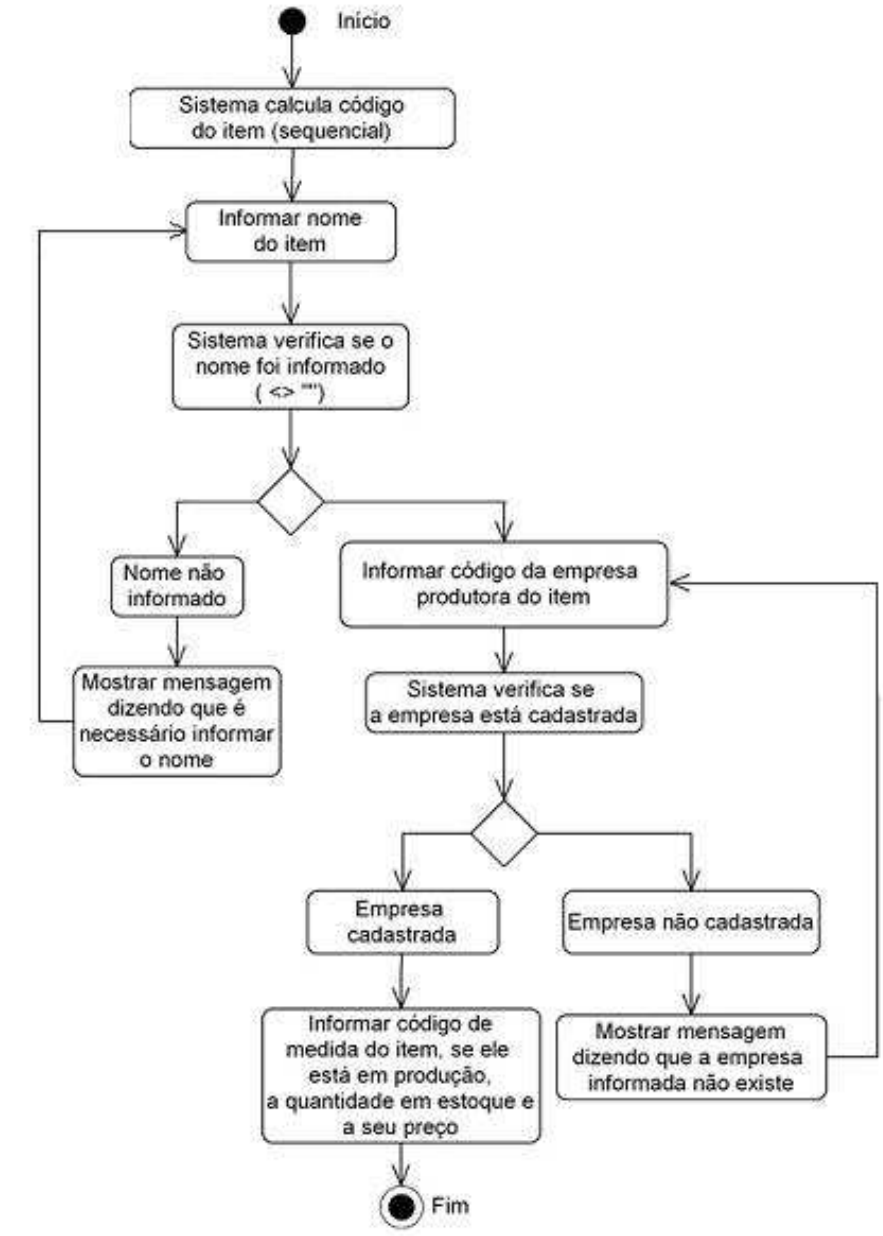
Cadastro de Transportadores



Cadastro de Vendedor



Cadastro de Item



9. DICIONÁRIO DE DADOS

Bairro – Tabela que contém informações sobre bairros cadastrados.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_bairro**	**INTEGER**	NN	Código do bairro, Único, maior que 0
	bai_nome	VARCHAR(25)	NN	Nome do Bairro

Cidade – Tabela que contém informações sobre cidades cadastradas.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_cidade**	**INTEGER**	NN	Código da cidade, Único, maior que 0
	cid_nome	VARCHAR(30)	NN	Nome da Cidade

Cliente – Tabela que contém dados pessoais dos clientes que realizam pedidos.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cnpj**	**VARCHAR(18)**	NN	CNPJ do Cliente, Único, formato “99.999.999/9999-99”
FK	Rua_cod_rua	INTEGER	NN	Código da rua, maior que 0
FK	Bairro_cod_bairro	INTEGER	NN	Código do bairro, maior que 0
FK	Cidade_cod_cidade	INTEGER	NN	Código da cidade, maior que 0
FK	Estado_cod_estado	INTEGER	NN	Código do estado, maior que 0
FK	Pais_cod_pais	INTEGER	NN	Código do país, maior que 0
	cli_nome	VARCHAR(45)	NN	Nome do Cliente
	fone	VARCHAR(13)		Telefone do Cliente, formato “(99)9999-9999”
	email	VARCHAR(45)		E-mail do Cliente
	end_num	INTEGER		Número da casa / prédio
	end_cep	VARCHAR(9)	NN	CEP formato “99999-999”

Empresa - Tabela que contém informações sobre a empresa produtora do Item.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_empresa**	**INTEGER**	NN	Código da Empresa, Único, maior que 0
FK	Rua_cod_rua	INTEGER	NN	Código da Rua, maior que 0
FK	Bairro_cod_bairro	INTEGER	NN	Código do bairro, maior que 0
FK	Cidade_cod_cidade	INTEGER	NN	Código da cidade, maior que 0
FK	Estado_cod_estado	INTEGER	NN	Código do estado, maior que 0
FK	Pais_cod_pais	INTEGER	NN	Código do país, maior que 0
	emp_nome	VARCHAR(45)	NN	Nome da Empresa
	end_num	INTEGER		Número do estabelecimento
	end_cep	VARCHAR(9)	NN	CEP formato “99999-99”
	fone	VARCHAR(13)		Telefone da Empresa formato “(99)9999-9999”

Estado - Tabela que contém informações sobre os estados cadastrados.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_estado**	**INTEGER**	NN	Código do Estado, Único, maior que 0
	est_nome	VARCHAR(30)	NN	Nome do Estado

Pais - Tabela que contém informações sobre países cadastrados.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_pais**	**INTEGER**	NN	Código do País, Único, maior que 0
	pais_nome	VARCHAR(25)	NN	Nome do País

Pedido - Tabela que contém dados de todos os pedidos efetuados pelo vendedor ao cliente.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**num_pedido**	**INTEGER**	NN	Número do Pedido, Único, maior que 0
FK	Vendedor_cpf	VARCHAR(14)	NN	CPF do Vendedor, formato “999.999.999-99”

FK	Transportador_cnpj	VARCHAR(18)	NN	CNPJ do Transportador, formato “99.999.999/9999-99”
FK	Cliente_cnpj	VARCHAR(18)	NN	CNPJ do Cliente, formato “99.999.999/9999-99”
	dt_ini_ped	DATE	NN	Data de abertura do Pedido, formato DD/MM/AAAA
	qtd_total	FLOAT		Quantidade Total do Pedido, com 2 casas decimais
	vlr_total	FLOAT		Valor Total do Pedido, com 2 casas decimais
	dt_fim_ped	DATE	NN	Data do encerramento do Pedido, formato DD/MM/AAAA
	situacao	VARCHAR(3)		CAN (cancelado), ATI (ativo), ENC (encerrado)
	obs	VARCHAR(100)		Observações do pedido

Restrições:

- Clientes não podem possuir mais de 5 pedidos em aberto.

PedProd - Tabela de relacionamento entre Pedidos e Produtos, ou seja, armazena quais são os produtos que estão num pedido e suas informações específicas.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	Pedido_num_pedido	INTEGER	NN	Número do Pedido, maior que 0
FK	Produto_cod_produto	INTEGER	NN	Código do Produto, maior que 0
	seq	INTEGER	NN	Seqüência do produto no pedido
	qtd	FLOAT	NN	Quantidade do Produto, com 2 casas decimais
	vlr_tot	FLOAT	NN	Valor total do Produto (qtd * Produto_preco), com 2 casas decimais

Restrições:

- A quantidade do item no pedido não poderá exceder a quantidade do mesmo em estoque.

Produto - Tabela que contém informações sobre os produtos fabricados pela Empresa.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_produto**	**INTEGER**	NN	Código do Produto, Único, maior que 0
FK	Empresa_cod_empresa	INTEGER	NN	Código da Empresa, maior que 0
	nome	VARCHAR(50)	NN	Nome do Produto
	cod_medida	VARCHAR(2)	NN	Código de Medida (pc, m2, un, cx)
	em_prod	BOOL	Yes	Produção ou não
	qtd_estoque	FLOAT	NN	Quantidade do produto em estoque
	preco	FLOAT	NN	Preço do Produto

Rua – Tabela que contém informações sobre ruas cadastradas para clientes, empresa, transportadores e vendedores.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cod_rua**	**INTEGER**	NN	Código da rua, Único, maior que 0
	rua_nome	VARCHAR(30)	NN	Nome da Rua

Transportador - Tabela que contém dados sobre o transportador de um pedido.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cnpj**	**VARCHAR(18)**	NN	CNPJ do Transportador, Único, formato “99.999.999/9999-99”
FK	Rua_cod_rua	INTEGER	NN	Código da Rua, maior que 0
FK	Bairro_cod_bairro	INTEGER	NN	Código do bairro, maior que 0
FK	Cidade_cod_cidade	INTEGER	NN	Código da cidade, maior que 0
FK	Estado_cod_estado	INTEGER	NN	Código do estado, maior que 0
FK	Pais_cod_pais	INTEGER	NN	Código do país, maior que 0
	transp_nome	VARCHAR(30)	NN	Nome do Transportador
	fone	VARCHAR(13)		Telefone do Transportador, formato

				“(99)9999-9999”
	email	VARCHAR(45)		E-mail do Transportador
	end_num	INTEGER		Número da casa / estabelecimento
	end_cep	INTEGER		CEP

Vendedor - Tabela que contém os dados pessoais sobre o vendedor de um pedido.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**cpf**	**VARCHAR(14)**	NN	CPF do vendedor, Único, formato “999.999.999-99”
FK	Rua_cod_rua	INTEGER	NN	Código da rua, maior que 0
FK	Bairro_cod_bairro	INTEGER	NN	Código do bairro, maior que 0
FK	Cidade_cod_cidade	INTEGER	NN	Código da cidade, maior que 0
FK	Estado_cod_estado	INTEGER	NN	Código do estado, maior que 0
FK	Pais_cod_pais	INTEGER	NN	Código do país, maior que 0
	vend_nome	VARCHAR(45)	NN	Nome do Vendedor
	fone	VARCHAR(13)		Telefone do Vendedor, formato “(99)9999-9999”
	email	VARCHAR(45)		E-mail do Vendedor
	end_num	INTEGER		Número da casa / prédio
	end_cep	INTEGER		CEP

Rua_Bairro – Tabela que indica a relação entre ruas e bairros, ou seja, quais ruas pertencem aos bairros.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	cod_rua	INTEGER	NN	Código da rua
FK	cod_bairro	INTEGER	NN	Código do bairro

Bairro_Cidade – Tabela que indica a relação entre bairros e cidades, ou seja, quais bairros pertencem as cidades.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	cod_bairro	INTEGER	NN	Código do bairro
FK	cod_cidade	INTEGER	NN	Código da cidade

Cidade_Estado – Tabela que indica a relação entre cidades e estados, ou seja, quais cidades pertencem aos estados.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	cod_cidade	INTEGER	NN	Código da cidade
FK	cod_estado	INTEGER	NN	Código do estado

Estado_País – Tabela que indica a relação entre estado e país, ou seja, quais estados pertencem aos países.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	cod_estado	INTEGER	NN	Código do estado
FK	cod_pais	INTEGER	NN	Código da país

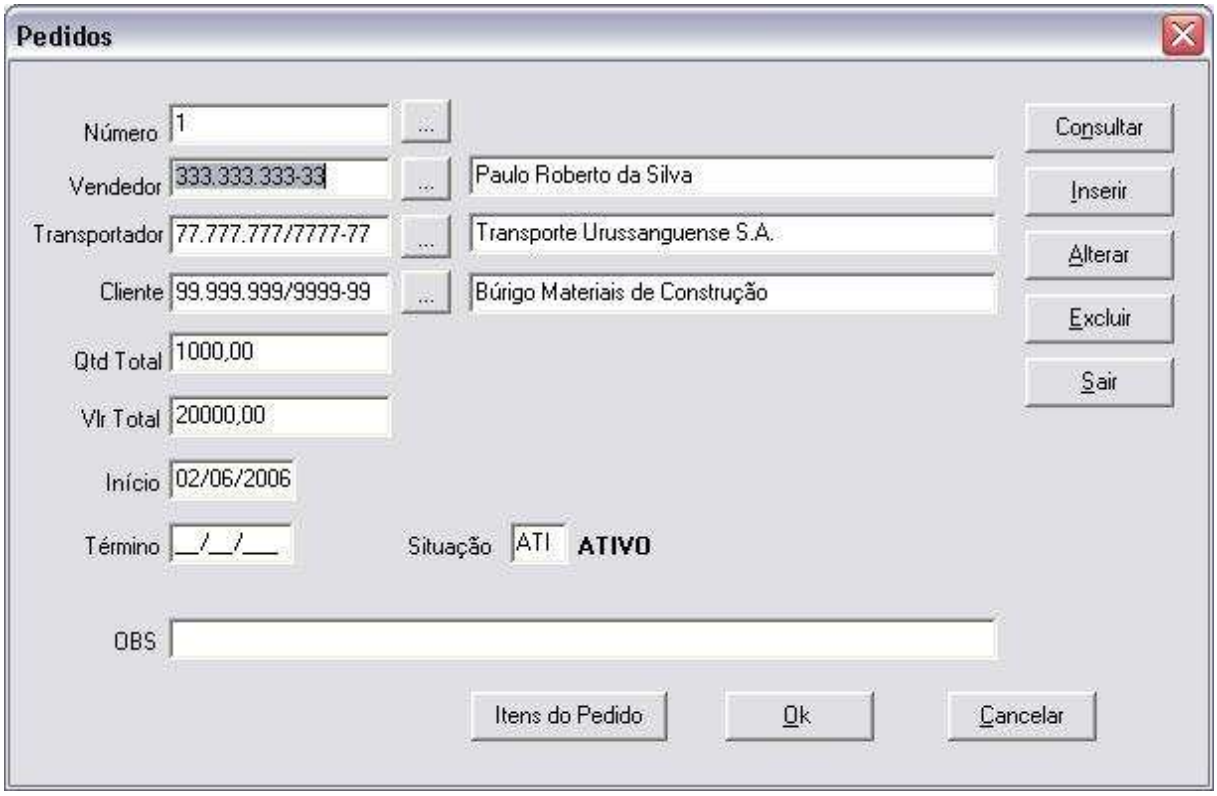
10. INTERFACE

10.1 – Autenticação

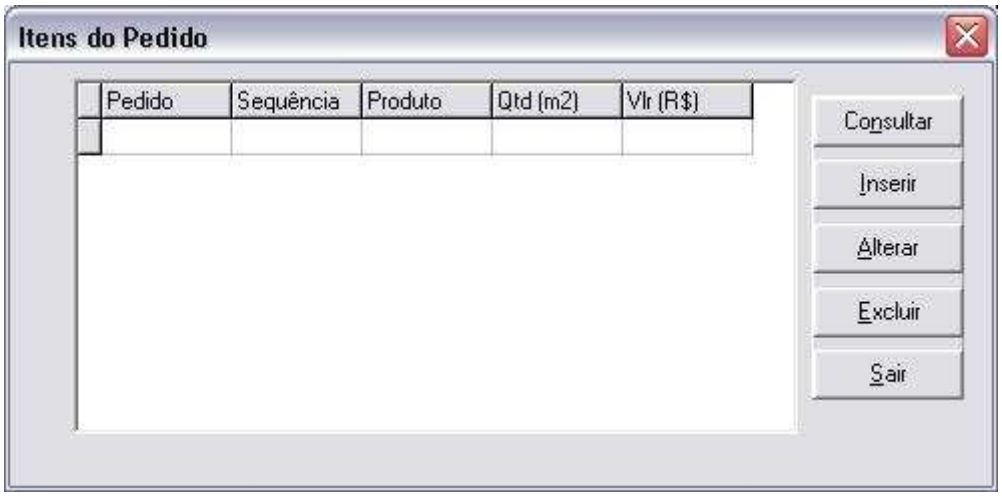
10.2 – Início

Obs.: Os relatórios ainda não foram discutidos com o cliente, porém já estão previstos na interface.

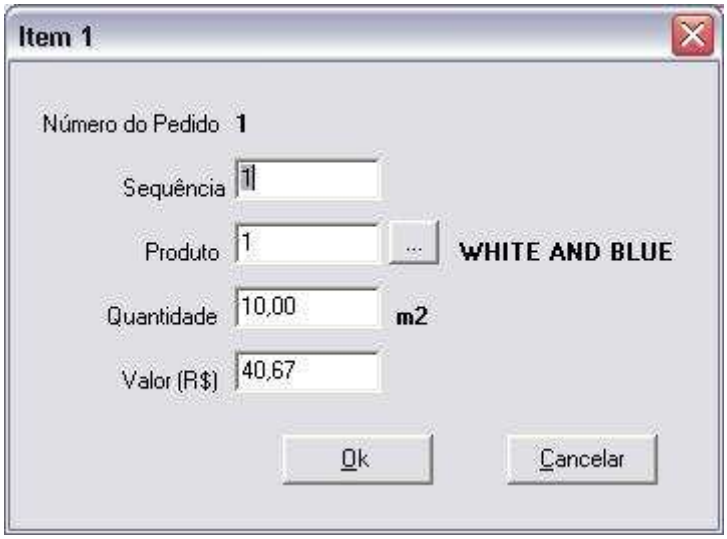
10.3 – Pedidos



10.4 – Lista Itens dos Pedidos



10.5 – Manutenção Itens dos Pedidos



11. AVALIAÇÃO

11.1. Periodicidade

- No primeiro mês será feito um levantamento de todos os dados necessários para que se possa inicializar o projeto.
- Uma vez por mês será feita uma reunião com o cliente afim de lhe repassar o andamento do projeto, observar pendências ou necessidades de mudança.
- Uma vez por mês o cliente irá até o desenvolvedor para teste do sistema.

11.2. Correção

Cada vez que algum dos pontos previstos na periodicidade não forem atendidos pelo cliente, isso lhe acarretará uma multa (acréscimo) de 2% no valor da implementação.

Da mesma forma, cada vez que algum ponto não for atendido pelo contratado, isso lhe acarretará uma multa (desconto no software) de 2% no valor da implementação.

O levantamento de dados que era pra ser feito no primeiro mês, levou um mês e meio. Como isso depende tanto do cliente quanto do contratado, multas não foram aplicadas a nenhuma das partes.

12. PROJETO DE CUSTO

12.1. Recursos

Os recursos dizem respeito à infra-estrutra que a empresa ainda não possui e deverá ser adquirido.

12.2. Cronograma de Execução

- Aquisição de materiais:

Período / Atividade	Set 06	Out 06	Nov 06	Dez 06	Jan 07	Fev 07	Mar 07
Computadores							
4 mesas para computador							
1 Impressora							
1 mesa para impressora							
4 cadeiras para computador							
1 Data Show							
Rede com cabeamento par trançado da sala							

12.3. Cronograma de Desembolso (R$)

Período / Atividade	Set 06	Out 06	Nov 06	Dez 06	Jan 07	Fev 07	Mar 07
4 Computadores	2500,00		2500,00		2500,00		2500,00
4 mesas para computador	75,00		75,00		75,00		75,00
1 Impressora				400,00			
1 mesa para impressora				75,00			
4 cadeiras para computador	70,00	210,00					
1 Data Show						3000,00	
01 Switch 24 portas			280,00				
01 Modem para acesso ADSL		190,00					
01 Bracket 10 U				320,00			
01 Patch Panel 24 P		355,00					
02 Guias de cabo		34,00					
01 Régua de Tomadas (com 6 tomadas)		45,00					
06 Conectores RJ 45 fêmea				90,00			
06 Patch cord 2,5 m		48,00					
06 Patch cord 1,5 m		45,00					

Cabo Multilan 24 AWG 4 pares				3,40			
Implementação	1500,00	1000,00	100,00	1000,00	500,00		
Treinamento de Usuários							1000,00
Total	4145,00	1927,00	2955,00	1884,00	3075,00	3000,00	3575,00

Total Geral: R$ 20565,40.

Observações Importantes:

Após Agosto de 2007, o valor de alterações não previstas será cobrado por hora: R$150,00.

Caso a empresa contratante desejar realizar o treinamento de seus usuários por si própria, não precisará arcar custos com a empresa desenvolvedora.

13. CRONOGRAMA GERAL

Período / Atividade	Set 06	Out 06	Nov 06	Dez 06	Jan 07	Fev 07	Mar 07	Abr 05	Maio 07	Jun 07	Jul 07	Ago 07
Últimas decisões junto ao Cliente												
Implementação												
Reuniões com Cliente												
Montagem da Rede												
Implantação												
Testes												
Treinamento de Usuários												
Acompanhamento												

Obs: O suporte será dado ao cliente em prazo indeterminado, enquanto o software estiver sendo utilizado.

14. CONCLUSÃO

Hoje em dia não raras são as vezes em que ouvimos dizer: “Tempo é dinheiro!”. Por causa disso, muitas das empresas produtoras de software, recusam-se a dedicar tempo no planejamento dos seus produtos. Isso é um grande erro, pois o ato de planejar e analisar as coisas antes de serem feitas, pode prevenir que algo venha a dar errado futuramente, envolvendo custos para sua solução e ainda acarretando um desperdício maior de tempo do que o previsto.

Tratando-se do software aqui proposto, percebe-se a real necessidade e importância do mesmo para uma empresa que trabalha com produção e venda de um produto, nesse caso revestimentos cerâmicos, pois de nada vale fabricar um produto se uma empresa não puder vendê-lo, e para a venda é preciso de controle.

O objetivo do trabalho foi alcançado, pois prevê o desenvolvimento, que já foi iniciado, e futuras instalações do software de controle de vendas de uma empresa de revestimentos cerâmicos.

Vale ressaltar que este software foi previamente planejado para começar a ser desenvolvido e que até então não foram encontradas dificuldades.

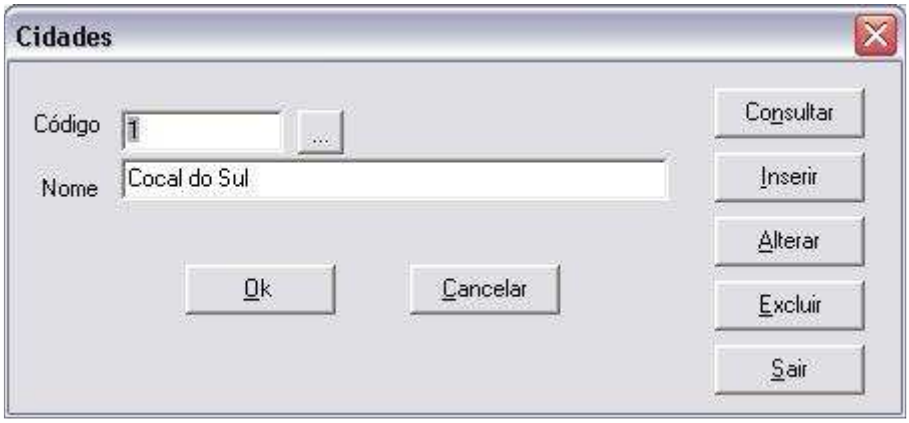
Fica fácil perceber que com um planejamento adequado, poder-se-á economizar não só tempo, mas também dinheiro nas partes posteriores do desenvolvimento de qualquer software. Daí surge a seguinte pergunta: Melhor economizar tempo e gastar dinheiro, ou economizar dinheiro e gastar tempo?

15. ANEXOS

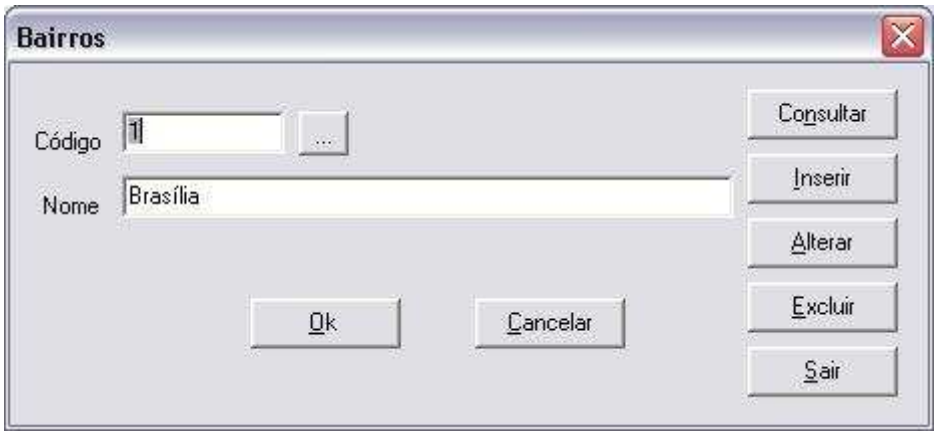
Países

Estados

Cidades



Bairros



Ruas

Relacionamento Estados e Países

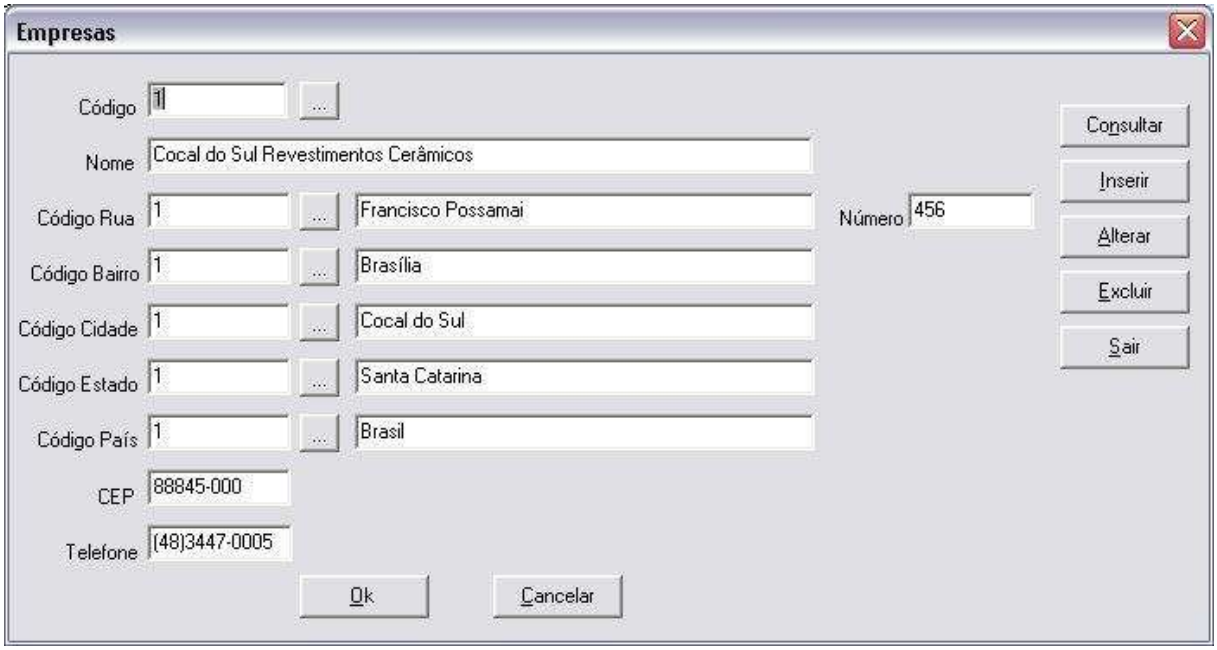
Relacionamento Cidades x Estados

Relacionamento Bairros x Cidades

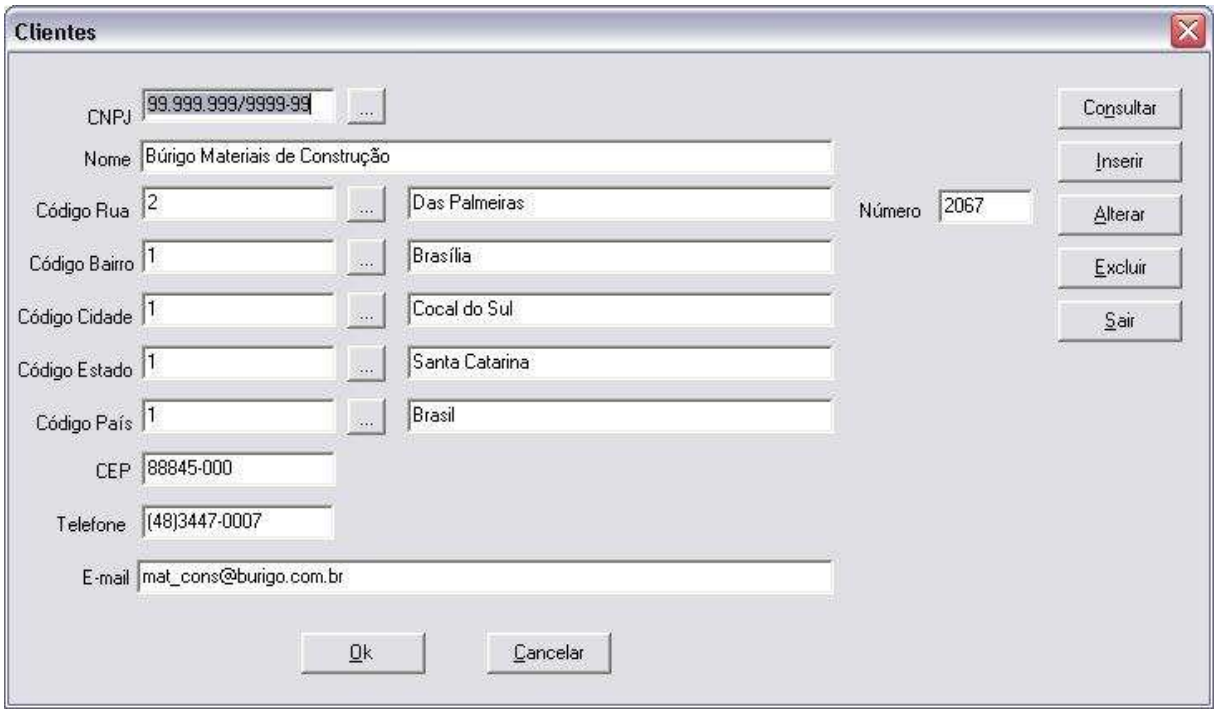
Relacionamento Ruas x Bairros


Ruas e Bairros
Código Rua
1
...
FRANCISCO POSSAMAI
Código Bairro
1
...
BRASÍLIA
Ok
Cancelar
Consultar
Inserir
Alterar
Excluir
Sair

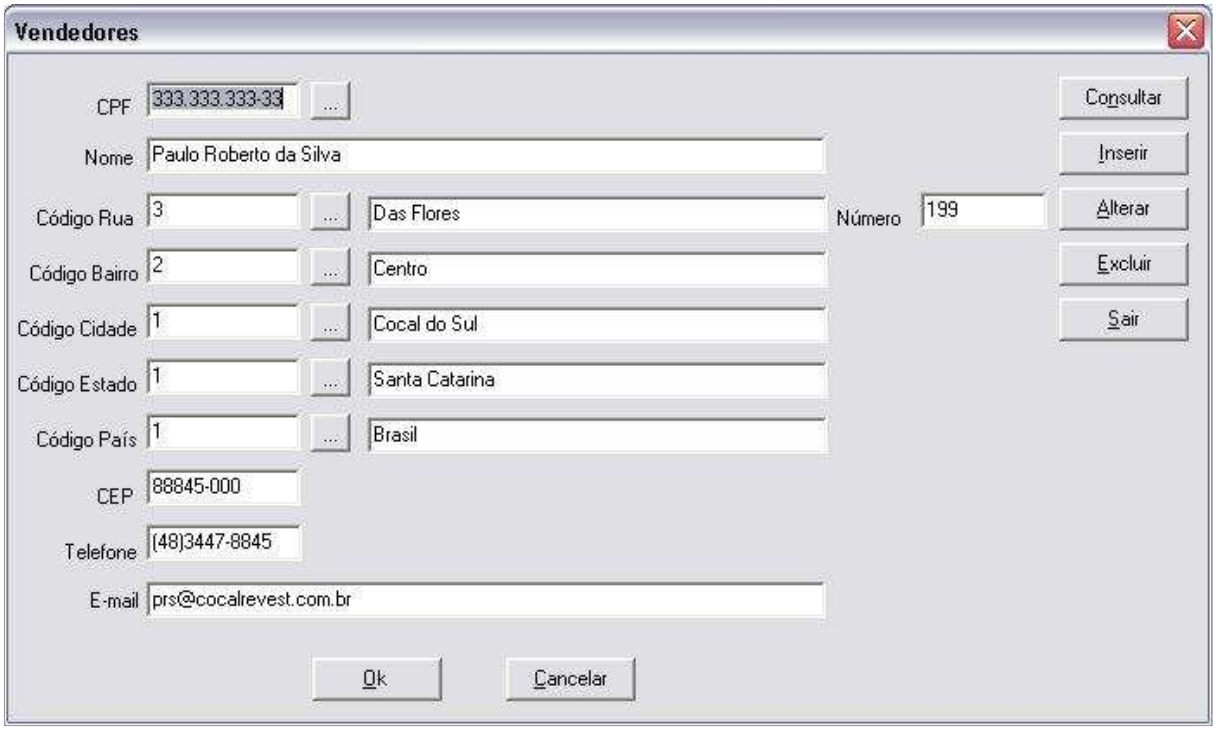
Empresas



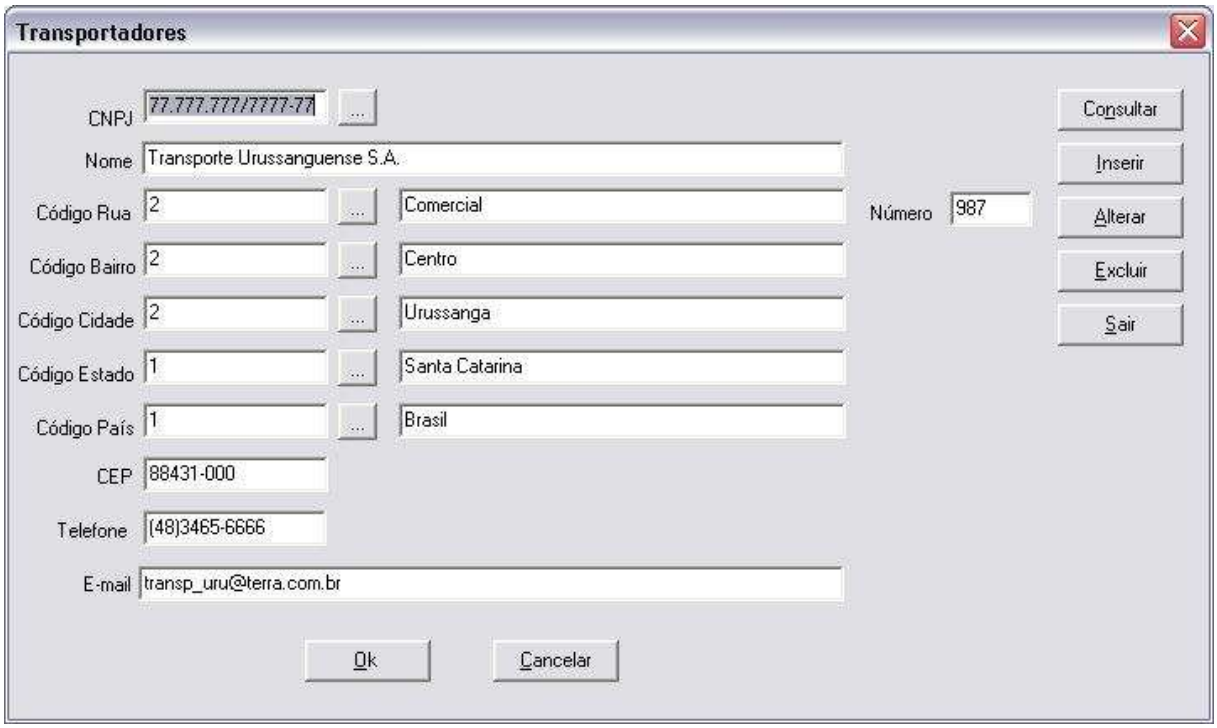
Clientes



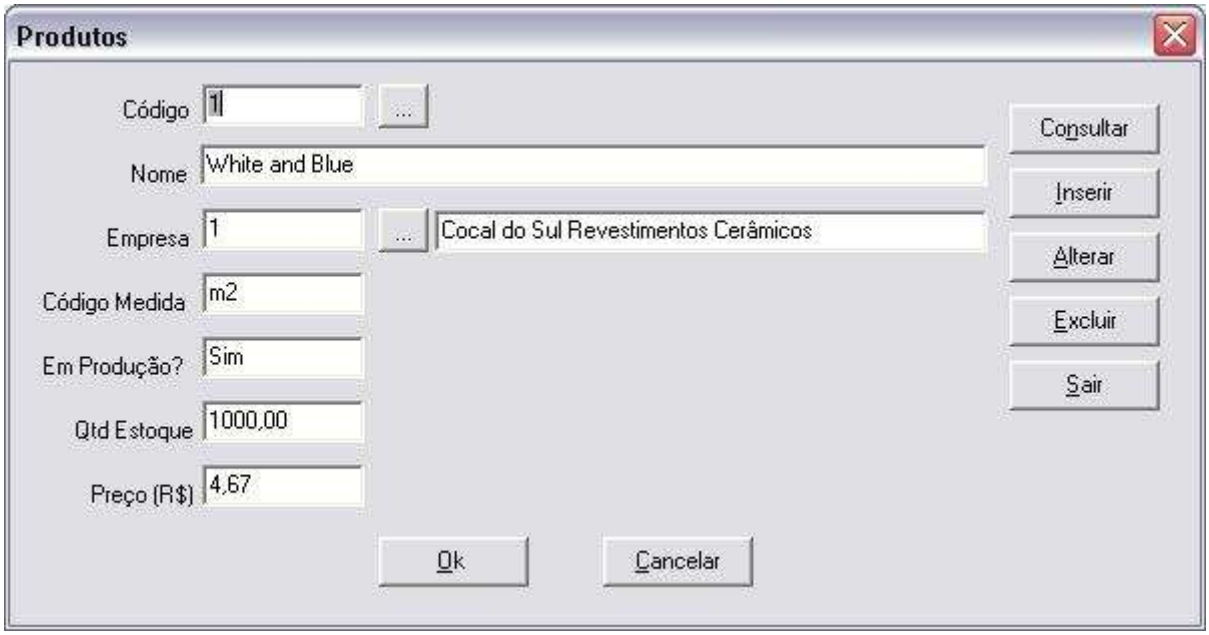
Vendedores



Transportadores



Produtos


Produtos
Código
1
...
Nome
White and Blue
Empresa
1
...
Cocal do Sul Revestimentos Cerâmicos
Código Medida
m2
Em Produção?
Sim
Qtd Estoque
1000,00
Preço (R$)
4,67
Consultar
Inserir
Alterar
Excluir
Sair
Ok
Cancelar