UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO ROSA

ESTUDO DE MODELAGEM LÓGICA DE SISTEMAS DA ÁREA DA SAÚDE, INDUSTRIAL E COMERCIAL, PARA MANIPULAÇÃO EM UM REPOSITÓRIO EM BANCO DE DADOS OBJETO-RELACIONAL

CRICIÚMA, JULHO DE 2011

TIAGO ROSA

ESTUDO DE MODELAGEM LÓGICA DE SISTEMAS DA ÁREA DA SAÚDE, INDUSTRIAL E COMERCIAL, PARA MANIPULAÇÃO EM UM REPOSITÓRIO EM BANCO DE DADOS OBJETO-RELACIONAL

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

CRICIÚMA, JULHO DE 2011

TIAGO ROSA

Estudo de Modelagem Lógica de Sistemas da Área da Saúde, Industrial e Comercial, para Manipulação em um Repositório em Banco de Dados Objeto-Relacional.

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

Profª. MSc. Ana Cláudia Garcia Barbosa (UNESC)

Prof. MSc. Gustavo Bisognin (UNESC)

Dedico este trabalho ao meu Pai Zeli e minha Mãe Vanilda.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela família que me deu, pelas oportunidades que me ofereceu, pelas pessoas que colocou no meu caminho, por estar sempre me concedendo graças e me dar força necessária nos momentos difíceis, especialmente quando achei que não conseguiria concluir este trabalho.

Graças a ele posso considerar-me uma pessoa privilegiada.

Agradeço à minha Mãe, Vanilda Formanski Rosa, meu porto seguro, pelo seu constante apoio, amor e dedicação, que nas horas difíceis estava ali presente, transmitindo força e coragem.

Agradeço a meu Pai, Zeli Mario Rosa, meu chão incontestável, pelo amor e pelo seu olhar silencioso que me transmitia: “Você vai conseguir filho!”. Mesmo sabendo dos problemas e dificuldades que eu estava passando.

Agradeço ao meu irmão por aturar meu estado constantemente irritado.

Agradeço ao meu Professor Msc. Paracelso de Oliveira Caldas, pelo incentivo, orientação competente e relevantes ensinamentos repassados nestes anos de convivência acadêmica. Agradeço-lhe também pelo exemplo de profissionalismo, dedicação, pelas sugestões e críticas enriquecedoras, pela disponibilidade, paciência e amizade.

Agradeço também a todos que caminharam comigo nestes anos de estudos e dificuldades na vida acadêmica. Sem sombra de dúvida, um grande aprendizado para a vida.

“Quando mergulhei no futuro, tão longe quanto o olho humano pode alcançar, tive a visão do mundo e de toda a maravilha em que poderia se tornar”

Tennyson

RESUMO

Diante do atual contexto competitivo no que se refere a desenvolvimento de *software*, a modelagem lógica de dados e a armazenagem em um banco de dados objeto-relacional, possui um diferencial significativo no mercado de *softwares*. Para tanto, uma forma de modelagem mais ágil, possibilitando a redução do tempo de implantação e custos relacionados ao projeto, torna-se de grande valia para o adquirente. Assim, diante da necessidade empresarial, este estudo tem por objetivo a criação de um protótipo para manipular esta modelagem, de uma forma estruturada em uma base de dados (repositório). Para tal, foram realizados estudos bibliográficos, em que se buscou estabelecer relações entre o marco teórico do trabalho e a realidade do protótipo a ser desenvolvido. Desta forma, os resultados do estudo evidenciam uma dificuldade na criação de um protótipo, sendo necessária a utilização de uma ferramenta externa, que possibilitasse ao desenvolvedor do *software*, executa as operações a partir de um banco de dados existente, uma vez que parte das informações e dos dados já estejam estruturados para utilização. Com isso o desenvolvedor de sistemas utilizará as informações existentes no banco de dados e fará as modelagens a partir destas informações para manutenção e desenvolvimento de aplicações. O Desenvolvimento deste protótipo foi possível com a utilização da ferramenta Astah Community, para a modelagem de dados, o Microsoft SQL Server Express como banco de dados e a linguagem de programação Visual FoxPro. Concluindo foi implementado a funcionalidade de manipulação de um modelo existente (criado pela ferramenta Astah), assim dando a possibilidade de gravar num banco de dados, para novas pesquisas.

Palavras-chave: Modelagem lógica. Repositório. Objeto-relacional.

ABSTRACT

In today's competitive environment with regard to software development, modeling and logical data storage in a database object-relational, has a substantial gap in the software market. For that, one way of modeling more agile, enabling the reduction of deployment time and costs related to the project, it is of great value to the acquirer.Thus, given the business need, this study aims to create a prototype model to handle this in a structured manner in a database (repository). To this end, studies were performed bibliographic, which sought to establish relationships between theoretical work and the reality of the prototype to be developed. Thus, the study results show a difficulty in creating a prototype, being necessary to use an external tool that would enable the developer of the software, performs the operations from an existing database, as part of the information and data are already structured to use. With this, the system developer will use the information in the database and make the modeling based on this information for maintenance and application development. The development of this prototype was made possible with the use of Community Astah tool for data modeling, Microsoft SQL Server Express as database and programming language Visual FoxPro conclusion was implemented functionality for manipulating an existing model (created by Astah tool), thus giving the possibility to record in a database for further research.

Keywords: Logic modeling. Repository. Object-relational.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BD	- Banco de Dados
BPMN	- *Business Process Modeling Notation*
CASE	- Engenharia de *Software* Auxiliada por Computador *(Computer-Aided Software Engineering)*
CMMI	- *Capacity Maturity Model Integration*
ER	- Engenharia de Requisitos
ODP	- *Open Distributing Processing*
OODBMS	- Sistema de Gerenciamento de Banco de Dados Orientados a Objetos
SGBD	- Sistema de Gerenciamento de Banco de Dados
SGBDOR	- Sistema de Gerenciamento de Banco de Dados Objeto-Relacional

SUMÁRIO

1 INTRODUÇÃO

A utilização da informática em constante evolução tem demonstrado ser o grande aliado nos processos de construção de conhecimento que envolve dados. Por sua vez o desenvolvimento de *software* vem sendo de crescente importância para sociedade. Por meio dele podemos utilizar os computadores nas mais diversas áreas do conhecimento humano gerando informações para análise.

Observa-se que muitas vezes o desenvolvimento de *software* é confundido com programação. No entanto, deve-se levar em consideração os métodos usados nesta área de conhecimento, iniciando-se pelo desenvolvimento das habilidades de raciocínio lógico, atrelados à área programação, estruturas de dados e análises.

Neste sentido, para cada problema encontrado necessita-se de maiores conhecimentos e habilidades, diante do tamanho e a complexidade do que se pretende resolver. Observa-se que para problemas de maior complexidade, necessita-se de métodos específicos para sua resolução. Métodos estes, que envolvem um estudo mais avançado do problema, ou seja, realiza-se uma análise, com estudos detalhados para a solução mais adequada possível.

Identificando-se o problema deve-se fazer todo planejamento para a solução do mesmo, e ao longo da realização do trabalho, realiza-se um acompanhamento para a verificação de prazos, custos e qualidade do trabalho e objetivo a ser alcançado.

Com o intuito de melhorar a qualidade dos produtos de *software* e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de *Software*. A qual trata de aspectos relacionados à consignação de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de *software*. Como em outras áreas, o problema a ser tratado deve ser analisado e dividido em partes menores, utilizando o conceito “dividir

para conquistar". Para cada uma dessas partes, uma solução deve ser elaborada. Esta solução pode ser auxiliada com as técnicas e metodologias da engenharia de *software*, especificamente falando na engenharia de requisitos.

Para atingir o êxito no desenvolvimento, muitas vezes é impossível gerir o desenvolvimento de *software* de maneira individual. Existe uma equipe de trabalho nos bastidores, dispensando um esforço para concluir sua implantação, sendo coordenada e acompanhada, para que possa atingir sua qualidade esperada. Com intuito de facilitar este esforço, a equipe poderá contar com um banco de informações que será de suma importância, tanto para o custo final do projeto quanto ao tempo de implantação.

Diante do exposto, este trabalho analisa e identifica modelagens lógicas já existentes testadas e depuradas em Sistemas nas Áreas de Saúde, Industrial e Comercial, para Manipulação em um Repositório em Banco de Dados Objeto-relacional.

1.1 OBJETIVO GERAL

Armazenar os modelos lógicos de três tipos de sistemas nas respectivas áreas (saúde, industrial e comercial), a fim de, desenvolver um protótipo para manipular esta modelagem, de uma forma estruturada em uma base de dados (repositório).

1.2 OBJETIVOS ESPECÍFICOS

Para a obtenção do resultado esperado do objetivo geral, têm-se os seguintes objetivos específicos:

a) pesquisar e entender três tipos de *softwares* existentes no mercado de *software* nas áreas da saúde, indústria e comércio;

b) identificar e compreender a modelagem lógica dos *software*s estudados;

c) catalogar estas informações utilizando a linguagem de modelagem UML;

d) fornecer estas informações por meio de um protótipo, para futuras consultas.

1.3 JUSTIFICATIVA

O tema abordado reveste-se de importância nos assuntos relacionados ao desenvolvimento de *software*. Conforme Pressman (2006, p. 17), "a Engenharia de *Software* é a criação e a utilização de sólidos princípios de engenharia a fim de obter *softwares* econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais". Esta qualidade engloba fatores que são de importância para o desenvolvimento do projeto, sendo elas, a funcionalidade, confiabilidade, usabilidade, eficiência, portabilidade e manutenção. Por sua vez, estas variáveis são responsáveis pela qualidade do *software*. Neste sentido a engenharia deve conciliar a qualidade ao prazo estimado de entrega, atendendo a real necessidade do cliente. Caso o desenvolvedor não cumpra, com os prazos e com as estimativas previstas, o resultado gera o descontentamento do cliente e aumento de custo.

Analisando os processos de uma implantação citado na definição do problema para um desenvolvimento de *software*, observa-se certo grau de dificuldade do desenvolvedor em cumprir seus prazos. Nota-se então que os engenheiros perdem tempo para estudar a necessidade do cliente, este tempo é empregado no levantamento de dados, na análise de requisitos, ou seja, é gasto um tempo excessivo na coleta de informação e muitas vezes não conseguem compreender o propósito do *software* podendo resultar em um sistema falho.

Os engenheiros de *software* estão preocupados com os processos atuais de *software*s, onde a necessidade é entender, avaliar, controlar, aprender, comunicar e melhorar na qualidade do produto, e com este processo acaba-se perdendo tempo relevante,

influenciando no prazo de entrega do *software*, no custo final do projeto, como também na qualidade do *software.*

Objetivando reduzir o tempo e os custos das futuras implantações esta pesquisa tem como princípio o estudo de três tipos de sistemas (saúde, indústria e comércio) atuantes no mercado de *software*, onde será abstraída a idéia central de seus principais módulos. Estas informações serão estudadas e catalogadas para análise, logo após será realizada uma modelagem lógica destas informações e seu armazenamento em um repositório. Estas informações estarão disponíveis para consultas e uso, auxiliando os engenheiros de *software* aproveitando-as no desenvolvimento de novas aplicações, reduzindo o tempo de aquisição do conhecimento. Por meio das necessidade do cliente o engenheiro terá estas modelagens como embasamento para implementar o *software* utilizando os requisitos da modelagem proposta, economizando tempo e dinheiro.

Tendo como enfoque os problemas referentes à engenharia de *software*, e os conceitos encontrados na literatura sobre o aprendizado junto ao cliente. E analisando a relevância do tema, voltado para a área comercial, industrial e da saúde esta pesquisa apresenta como proposta a identificação e modelagem lógica de aplicações já existentes, aprovadas e consolidadas no mercado (utilizando métodos e padrões homologados por organizações Internacionais de padrões) para o seu armazenamento e manipulação em um banco de dados objeto-relacional.

1.4 ESTRUTURA DO TRABALHO

O presente Trabalho tem como estrutura oito capítulos organizados. Onde primeiro mostra a introdução, os objetivos, justificativa e estrutura do projeto proposto. O

segundo capítulo tem com foco mostrar conceitos relacionados à engenharia de *software*, suas características, seus processos, seus modelos o qual será aplicado ao referente projeto. No terceiro capítulo intitula a prática da engenharia de requisitos e a modelagem lógica de dados. No quarto capítulo o banco de dados relacional é o tema a ser mostrado. No quinto capítulo elenca-se os tipos de *softwares* objeto de estudo. Posteriormente no capítulo sexto, as ferramentas utilizadas em uma sintética apresentação. No sétimo capítulo os trabalhos correlatos são abordados sinteticamente. No oitavo capítulo é mencionado o trabalho juntamente com a metodologia, recursos utilizados. Finalizando com os elementos pós-textuais do trabalho.

2 ENGENHARIA DE *SOFTWARE*

Com o advento da evolução dos sistemas computadorizados, grande parte dos esforços e custos gerados, concentrava-se no desenvolvimento do *hardware* em razão das limitações encontradas na época. Neste contexto, com o passar do tempo, dominou-se os problemas e as atenções voltaram-se para o desenvolvimento de sistemas operacionais que contribuíram de forma significativa com a queda de preço do *hardware*. Posteriormente, com o crescimento em meio a necessidades, nasceu o termo "Engenharia de *Software*", e atualmente os preços dos equipamentos não acompanham a mesma tendência do desenvolvimento de *software*, o custo de um sistema informatizado, corresponde a uma percentagem cada vez maior. Neste sentido, a engenharia de *software* refere-se a princípios da engenharia que se ocupa dos aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção, dedicando-se aos problemas práticos da produção de *software*.

Segundo Sommerville (2003, p. 6), "engenharia de *software* é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema".

A Engenharia de *software* não dedica-se somente aos processos técnicos de desenvolvimento, mas também as atividades como o gerenciamento de projetos e a criação de ferramentas, métodos e teorias que dêem apoio à produção de sistemas de *software*, estabelecendo o uso de sólidos princípios de engenharia, obtendo-se um *software* economicamente confiável, funcionando eficientemente em máquinas reais".

O desenvolvimento de um sistema de *software* realiza-se por diversos métodos da Engenharia de *Software*. Para a equipe de desenvolvimento, cabe a tarefa de escolher dentre esses métodos qual ou quais se encaixam no projeto e aplicá-los da melhor forma possível,

seguindo os padrões da própria Engenharia de *Software*, aplicando-se uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção do *software*.

Compreende-se que a Engenharia de *Software* compõe-se de processos e modelos de desenvolvimento, que juntos possibilitam a criação de uma base mais sólida para a construção de um sistema de *software* de alta qualidade. Os modelos ou paradigmas de desenvolvimento de sistemas de *software* relacionam-se às representações do seu processo de criação. Os processos tratam-se dos procedimentos ou metodologias utilizadas para seu desenvolvimento.

2.1 *SOFTWARE* – PAPEL E CARACTERÍSTICAS

O acesso à informação tornou-se indispensável e neste meio, o mercado de *software* luta para garantir seu espaço. O *Software* trata-se de um elemento fundamental para o desenvolvimento econômico no século 21, quanto outros produtos foram importantes em épocas diferentes. Em meados dos anos 90, o *Hardware* perdia forças no custo das implantações dos sistemas. A partir deste momento o *software* começa a ganhar destaque no cenário de desenvolvimentos de *softwares*.

Os desenvolvedores visavam adequar seu *software* para atender os requisitos do *hardware*, pois o alto custo concentrava-se na aquisição de *hardware* com poder de processamento para executar as operações do *software* (PRESSMAN, 2006, p. 4-6).

Verifica-se então a evolução dos sistemas computadorizados mudando a sociedade industrial, para uma sociedade de informação, e para este objetivo ser alcançado, os *software*s e as redes de computadores tornam-se peças chaves para a difusão do conhecimento.

Desta forma, entende-se que a evolução dos sistemas fez com que os *software*s atendessem os requisitos dos *hardware*s, facilitando nas execuções das operações e disseminando conhecimento. Neste contexto, compreende-se por *software* como um sistema lógico e não físico que não se desgasta, mas, se deteriora que em sua maioria, a construção seja personalizada, não sendo montada a partir de componentes existentes, o desenvolvimento de *software* abrange um conjunto de três elementos fundamentais: métodos, ferramentas e procedimentos.

Desenvolvimento de *Software*		
- Elementos Fundamentais -		
Métodos	**Ferramentas**	**Procedimentos**
- Planejamento e estimativa de projeto	- Apoio aos métodos	- União entre métodos e ferramentas

Quadro 1. Desenvolvimento de *Software*
Fonte: Adaptado de Pressman, R. (2006, p. 17-18).

Os métodos definem "como fazer", envolve um planejamento e estimativa de projeto, o estudo de seus requisitos para o desenvolvimento de sua modelagem.

Emprega-se também a estrutura de processos, onde o engenheiro começa a conhecer o real propósito do *software*, qual o segmento a ser desenvolvido. Outros elementos como organização de dados, algoritmos e codificação são também métodos usados pelos engenheiros da área, pois com estas informações será possível realizar os devidos testes e consequentemente a manutenção do *software*.

As ferramentas definem-se como o apoio empregado aos métodos facilitando o a evolução do projeto. Conhecida também como Computer-Aided *Software* Engineering (CASE), responsável pelo enlace dos fatores *software*, *hardware* e banco de dados.

Os procedimentos caracterizam-se na união dos elementos métodos e ferramentas, o qual ditará a sequência em que os métodos serão aplicados, proporcionando um controle amplo em seu desenvolvimento (PRESSMAN, 2006, p. 17). Neste contexto, compreende-se

por características de *software*, a união de elementos essenciais para o desenvolvimento da modelagem necessária, para a criação de um sistema.

2.3 MODELOS DE DESENVOLVIMENTO DE *SOFTWARES*

Um processo de desenvolvimento de *software* trata-se um conjunto de atividades e resultados associados que auxiliam na produção de *software* (SOMMERVILLE, 2003, p. 36). Dentre as várias atividades associadas, pode-se citar a análise de requisitos e a codificação. Com o resultado do processo de *software* chega-se a um produto que reflete a forma como o processo foi conduzido. Verifica-se que diferentes organizações utilizam-se de processos diferentes para produção de um mesmo produto e alguns processos são mais adequados do que outros dependendo da aplicação. Embora existam vários processos para o desenvolvimento de *software*, existem atividades fundamentais comuns a todos eles. O Quadro 2 demonstra tais atividades.

Atividades para o desenvolvimento de *software*	
Especificação de *Software*	Definição das funcionalidades (requisitos) e das restrições do *software*.
Projeto e Implementação de *Software*	O *software* é produzido de acordo com as especificações.
Validação de *Software*	O *software* é validado para garantir as Funcionalidades.
Evolução de *Software*	O *software* precisa evoluir para continuar sendo útil ao cliente.

Quadro 2. Atividades para o Desenvolvimento de *Software*.
Fonte: Adaptado de Sommerville, I. (2003, p. 36).

A especificação de *software* refere-se à fase em que o desenvolvedor, define com o cliente as características do novo *software*. Em uma segunda fase no projeto e implementação de *software*, propõe-se modelos de diagramas, que posteriormente são implementados em alguma linguagem de programação. Na fase seguinte relaciona-se a validação do *software*, que se refere à garantia de que todas as funcionalidades especificadas

foram implementadas. A última atividade trata-se da evolução do *software*, o que assegura o cliente que de o *software* continuará sendo útil.

Ainda referindo-se a modelos de processos, encontram-se vários processos de *software* definidos na literatura da Engenharia de *Software*. Dentre os vários processos existentes, existem as metodologias tradicionais, que são orientadas a documentação, e as metodologias ágeis, que procuram desenvolver *software* com o mínimo de documentação.

Para Royce (1970 apud SOARES, 2004, p. 2),

> as metodologias tradicionais são também chamadas de pesadas ou orientadas a documentação. Essas metodologias surgiram em um contexto de desenvolvimento de *software* muito diferente do atual, baseado apenas em um *mainframe* e terminais burros.

A utilização desta metodologia, tornava o *software* planejado e documentado antes de ser implementado. Dentre as principais metodologias tradicionais, utiliza-se até os dias de hoje o Modelo Clássico.

Conforme Soares (2004, p. 5), nas "metodologias ágeis o enfoque nas pessoas e não em processos ou algoritmos". Com isso, gasta-se menos tempo com documentação e mais com a implementação, tornando-se adaptativas ao invés de serem preditivas.

Observa-se, que normalmente algumas organizações criam seu próprio processo ou adaptam à sua realidade. Os desenvolvedores de *software* implementam seu sistema sem usar nenhum processo. Isso ocorre porque o processo habitual não se adequou à necessidade da organização. Em particular, as organizações pequenas e médias não possuem recursos suficientes para adotar o uso de processos pesados. Por esta razão, muitas organizações não utilizam nenhum processo. O resultado desta falta de sistematização na produção de *software* é a baixa qualidade do produto final, além de dificultar a entrega nos prazos e custos predefinidos e inviabilizar a futura evolução. Um modelo de desenvolvimento corresponde a uma representação abstrata do processo que por fim direcionará até o alcance do objetivo, que

trata-se de um produto de baixo custo e alta qualidade. Na sequência, apresenta-se alguns modelos conhecidos e utilizados no desenvolvimento de *softwares*.

2.3.1 Modelo Clássico

O modelo clássico também chamado de Modelo em Cascata compreende em uma sistemática, sequencial ao desenvolvimento do *software*.

O modelo o cascata descreve um método de desenvolvimento linear e sequencial permitindo um controle departamental e gerencial, seguindo uma ordem estrita, sem qualquer sobreposição ou passos iterativos (PRESSMAN, 2006, 38-39).

Apresenta-se conforme Figura 01, o ciclo mais simples de desenvolvimento de *software*, estabelecendo o qual obedece a uma ordenação linear no que diz respeito à realização das diferentes etapas.

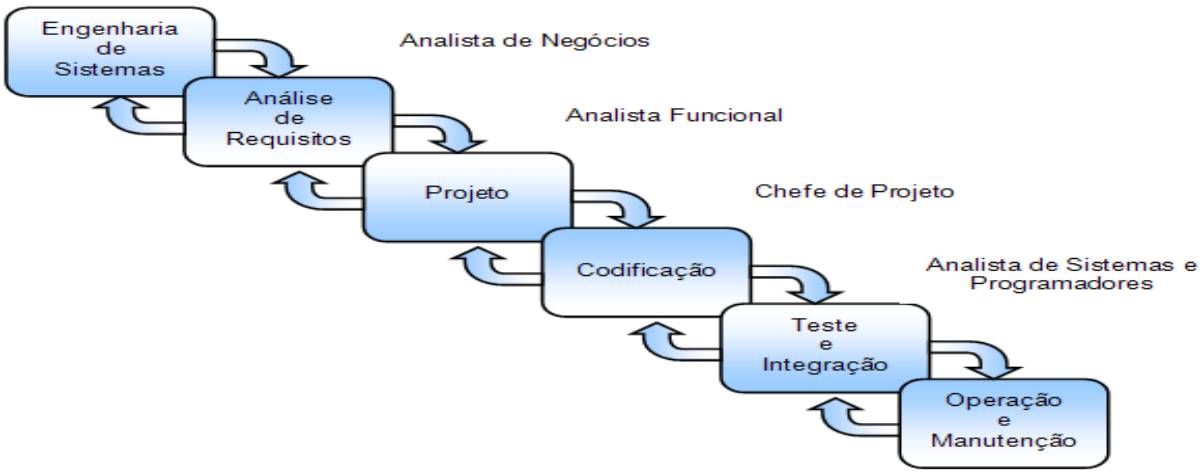


Figura 1. Ilustração do Modelo Clássico (Queda D'água).
Fonte: Adaptado de Pressman, R. (2006, p. 39).

No Ciclo de Vida Clássico utilizam-se conceitos de Engenharia de *Software*, a qual prevê atividade de verificação, validação e de controle de qualidade (PARREIRA

JUNIOR, 2011, p. 22). Trata-se de um paradigma que utiliza um método sistemático e sequencial em que o resultado de uma fase se constitui na entrada de outra fase, abrangendo as seguintes atividades ilustradas na Figura 1.

Atividades do Ciclo de Vida Clássico	
Engenharia de Sistemas	Quanto mais dados forem coletados em nível de sistema, menor será a probabilidade de haver "bugs" no sistema;
Análise de Requisitos	Saber o que o cliente quer que o *software* tenha, com relação a recursos, sendo documentados e revistos antes de começar a execução do projeto;
Projeto	Envolvem 4 atributos distintos do programa: estrutura de dados, arquitetura de *software*, detalhes procedimentais e caracterização de interface;
Codificação	O projeto deve ser transformado em programa, usando uma linguagem de programação, isto é, traduzido numa linguagem de máquina;
Teste e Integração	Testa-se os programas. A Integração consiste das junções das unidades e programas desenvolvidos concordando com o projeto;
Operação e Manutenção	O sistema é instalado e colocado em uso. A manutenção ocorre com a correção de erros não detectados, e na adição de novas características.

Quadro 3. Atividades do Ciclo de Vida Clássico.
Fonte: Adaptado de Parreira Junior, W. (2011, p. 22).

O ciclo de vida clássico refere-se ao paradigma mais antigo e o mais amplamente usado em Engenharia de *Software*, no entanto deve-se ater aos detalhes, pois corre-se riscos de não se atingir o objetivo final, tornando-se um produto de alto custo para a realização de correções.

2.3.2 Prototipação

O modelo de desenvolvimento baseado em um protótipo relaciona-se a uma versão preliminar do sistema apresentado ao usuário, onde ele fornece informações ao desenvolvedor, para que sejam realizadas adaptações e implementações, durante o projeto e desenvolvimento. Segundo Mazzola (2010, p. 1.10), “o objetivo da prototipação é um modelo

de processo de desenvolvimento que busca contornar limitações existentes no modelo cascata, eliminando a política de congelamento dos requisitos antes do projeto ou codificação".

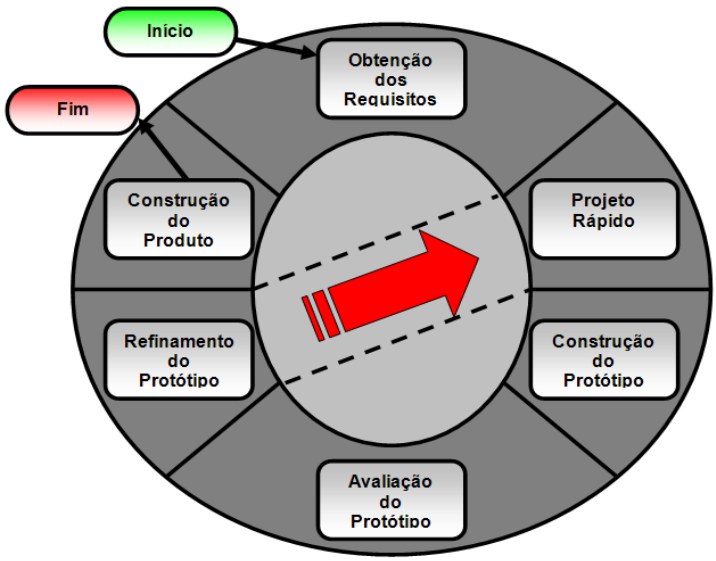


Figura 2. Prototipação.
Fonte: Adaptado de Pressman, R. (2006, p. 43).

De acordo com Oliveira L., (2009, p. 6), no modelo prototipagem (pura), "o desenvolvedor interage diretamente com o usuário, escutando seus pedidos e desenvolvendo, imediatamente, um protótipo do produto desejado". Com este modelo, o usuário utiliza o protótipo e fornece ao desenvolvedor outras novas informações que o levam a gerar atualizações, adaptações e implementações no *software*, durante o de projeto e desenvolvimento. Desta forma, observa-se que a prototipagem trata-se de uma técnica valiosa no levantamento de requisitos, tornando-se mais reais, facilitando o entendimento. Com isso, ao colocar o usuário na frente parte ou imitação do sistema, a prototipagem estimula os usuários a pensar e a estabelecer um diálogo sobre os requisitos (FALBO, 2011, p. 47).

Este modelo permite que diferentes tipos de protótipos possam ser desenvolvidos, conforme apresentado no Quadro 4.

Tipos de Protótipos Quanto:		
Arquitetura de Sistema	**Uso do Protótipo**	**Funcionalidades**
Protótipo não-operacional ou de interface	Protótipo descartável	Protótipo de características selecionadas
Protótipo operacional	Protótipo evolutivo	Protótipo completo

Quadro 4. Tipos de Protótipos.
Fonte: Adaptado de Falbo, R. (2011, p. 47-48).

Quanto às camadas da arquitetura de sistema que são efetivamente implementadas, o protótipo relaciona-se ao tipo operacional e não operacional.

O protótipo não-operacional ou de interface, apenas a camada de interface com o usuário é implementada, portanto, o sistema não faz nenhum processamento propriamente dito, já o protótipo operacional funciona como o sistema real deveria funcionar e implementar todas as camadas da arquitetura do sistema (FALBO, 2011, p. 47-48).

Referindo-se ao uso futuro do protótipo, como base para o sistema real ou não, um protótipo pode ser descartável, exploratório e não se pretende utilizá-lo como uma parte real do sistema a ser fornecido ou evolutivo, desenvolvido para se aprender mais sobre o problema e se ter a base de uma parte ou de todo o *software* a ser fornecido.

De acordo com o conjunto de funcionalidades provido pelo protótipo, ele pode ser de características selecionadas ou um protótipo completo. O protótipo de características selecionadas, apenas uma porção do sistema é implementada. Já o protótipo completo, apresenta todas as características do que se imagina ser o sistema real (FALBO, 2011, p. 48).

Compreende-se as particularidades das diferentes classificações de protótipos e que, no entanto podem ser combinadas, mas com seus devidos cuidados para que não se torne deficiente.

2.3.3 Modelo Espiral

O modelo cascata tornou-se a base da maior parte do desenvolvimento de projetos de *software*, durante anos. Já o modelo espiral sugerido posteriormente em 1988 por Barry Boehm, com a intenção de abranger as melhores características, no ciclo de vida clássico como da prototipação, acrescentando, ao mesmo tempo, um novo elemento, a análise de riscos que falta a esses paradigmas. O Quadro 5 descreve as quatro importantes atividades do modelo espiral, representadas por quatro quadrantes.

Modelo Espiral - Atividades	
Planejamento	Determinação dos objetivos, alternativas e restrições;
Análise de riscos	Análise de alternativas e identificação/resolução de riscos;
Engenharia	Desenvolvimento do produto no "nível seguinte";
Atualização feita pelo cliente	Avaliação dos resultados da engenharia.

Quadro 5. Atividades do Modelo Espiral.
Fonte: Adaptado de Lessa, R.; Lessa Junior, E. (2010, p. 5).

No modelo espiral mostra-se que as diferentes atividades são repetidas até uma decisão ser tomada e o documento de especificação de requisitos ser aceito. Compreende-se com este modelo, como sendo de uma abordagem "evolucionária" à engenharia de *software*, capacitando o desenvolvedor e o cliente a entender e reagir aos riscos em cada fase evolutiva.

O modelo espiral exige uma consideração direta dos riscos técnicos em todas as etapas do projeto e, se adequadamente aplicado, deve reduzir os riscos antes que eles se tornem problemáticos (Pressman, 2006, p. 45).

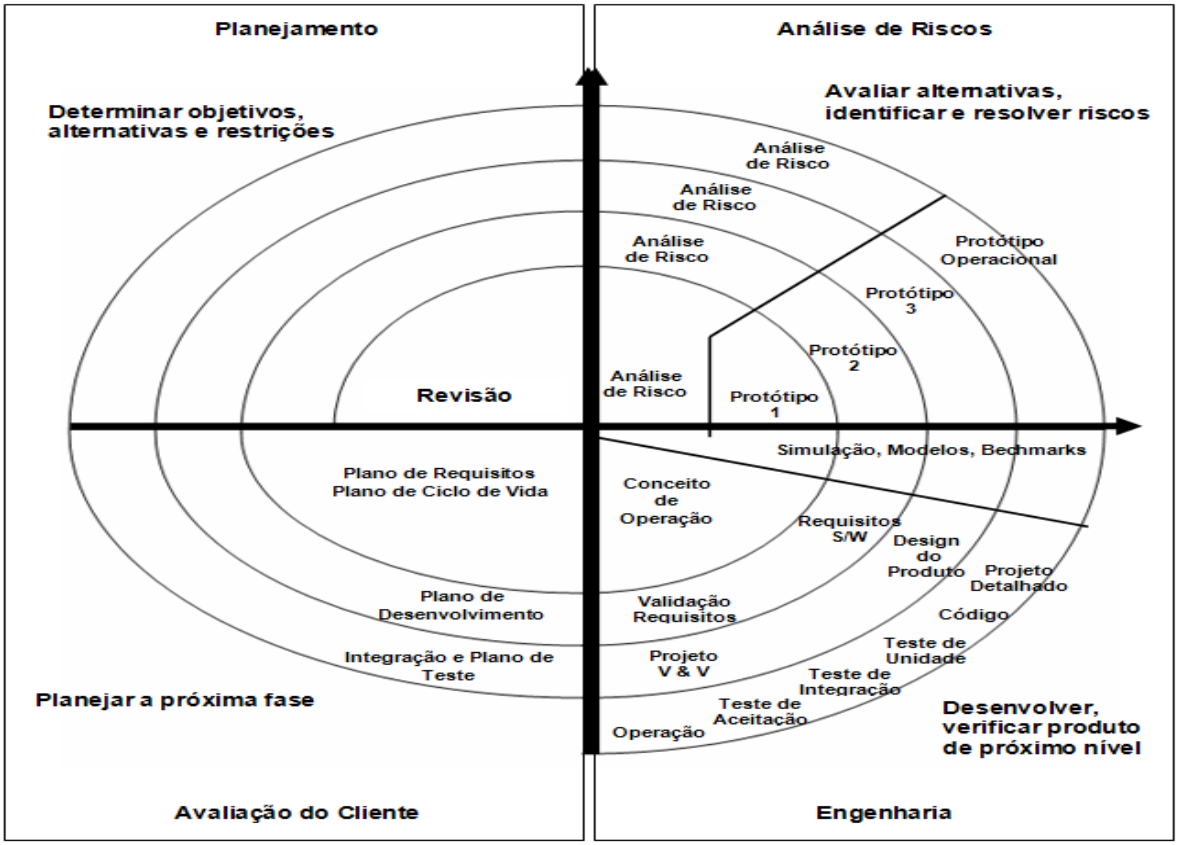


Figura 3. O Modelo Espiral.
Fonte: Adaptado de Sommerville, I. (2003, p. 45).

Na Figura 3 verifica-se, uma avaliação maior dos riscos de desenvolvimento de programa, bem como o protótipo obtido no passo corrente já resolve boa parte dos riscos ligados a desempenho, desta forma percebe-se a evolução segundo o modelo Queda D'água.

Observa-se que o elemento que conduz este processo trata-se da consideração sobre os riscos, o que permite a adequação a qualquer política de desenvolvimento. A continuidade do processo de desenvolvimento é definida como função dos riscos remanescentes, como por exemplo, a decisão se os riscos relacionados ao desempenho ou à interface são mais importantes do que aqueles relacionados ao desenvolvimento do programa (MAZZOLA, 2010, p. 1.12).

Considera-se uma característica importante, o encerramento de cada ciclo por uma atividade de revisão, onde todos os produtos do ciclo são avaliados, incluindo o plano para o

próximo ciclo. Abaixo representado no Quadro 6, menciona-se as atividades do ciclo do modelo espiral, e sucintamente a rotina de cada etapa do ciclo.

Atividades do Ciclo do Modelo Espiral		
Fases do Ciclo	**Atividades**	**Rotina**
	Elaboração	Elaborar objetivos, restrições e alternativas para entidades de *software*;
	Avaliação	Avaliar alternativas, restrições, e identificar as principais fontes de riscos;
	Desenvolvimento	Elaborar a definição das entidades de *software* em um projeto;
	Próxima Fase	Planejar o próximo ciclo. Abortar um projeto se apresentar riscos.

Quadro 6. Atividades do Ciclo do Modelo Espiral.
Fonte: Adaptado de Lessa, R.; Lessa Junior, E. (2010, p. 6).

Observou-se que o Modelo de Desenvolvimento em Espiral, surgiu com o objetivo de resolver os problemas dos projetos que seguem o fluxo que o modelo Cascata propõe. No entanto, no início do projeto torna-se difícil para o cliente especificar os requisitos. Desta forma, espera-se que os requisitos básicos estejam bem entendidos, e necessita-se que os detalhes sejam bem definidos. Em consequência com tempo, o produto evoluirá em versões mais completas.

2.3.4 Desenvolvimento Iterativo e Incremental

Necessita-se que um processo de desenvolvimento de *software* seja iterativo, isto é, ter várias iterações no tempo. Como também, necessita que seja incremental, isto é, gerar novas versões incrementadas a cada release. Para Silva, Souza e Dantas (2006, p. 18), “o modelo iterativo e incremental é uma alternativa para solução de problemas enfrentados no modelo cascata”.

Uma alternativa à abordagem em cascata é o processo iterativo e incremental, conforme mostrado na Figura 4.

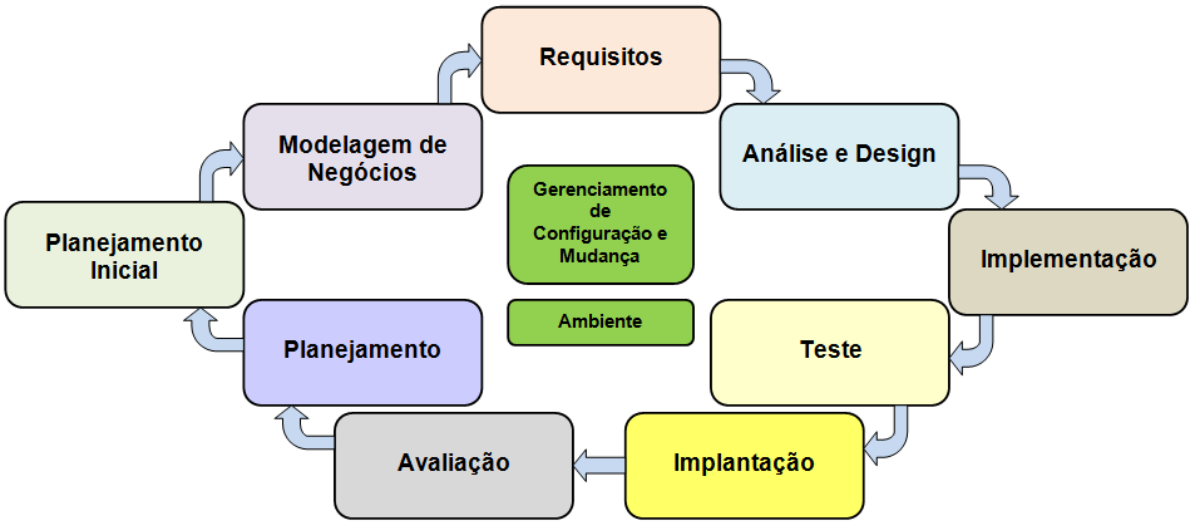


Figura 4. Abordagem Iterativa para Desenvolvimento.
Fonte: Adaptado de Bezerra, E. (2003).

Segundo Bezerra (2003), "um processo de desenvolvimento segundo essa abordagem divide o desenvolvimento de um produto de *software* em ciclos". Neste modelo, considera-se o desenvolvimento do *software* em ciclos iterativos, onde uma parcela dos requisitos passa por todas as etapas de desenvolvimento, semelhante ao modelo cascata.

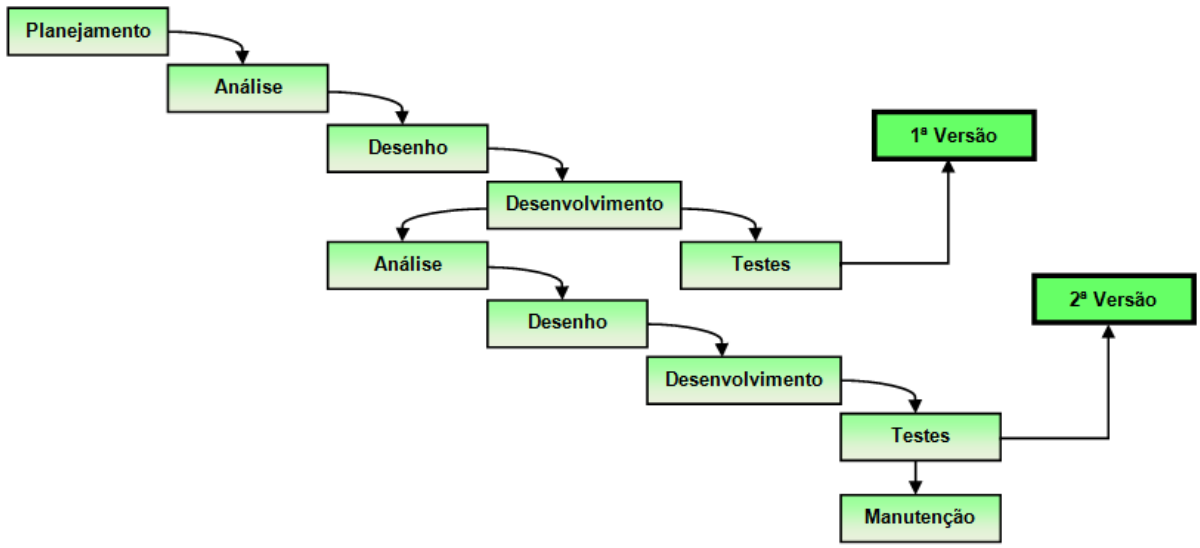


Figura 5. Desenvolvimento Iterativo e Incremental.
Fonte: Adaptado de Bezerra, E. (2003).

Na representação da Figura 05, visualiza-se que a cada ciclo de iteração têm-se uma nova versão do *software*, a qual será incrementada a cada novo ciclo.

Conforme Silva, Souza e Dantas (2006, p. 18), "ao final de cada ciclo de iteração pode-se ter uma versão do *software*". Neste sentido, a expectativa e a ansiedade do usuário em relação ao *software* diminuem. O modelo em questão possibilita desenvolvimento do *software* em módulos de acordo com as necessidades e características do projeto. Não descarta-se o planejamento formal neste modelo, as iterações e idéias permitem uma compreensão e refinamento a cada etapa, promovendo uma evolução gradativa.

A abordagem incremental e iterativa somente torna-se possível existindo um mecanismo para dividir os requisitos do sistema em partes, e cada parte alocada a um ciclo de desenvolvimento. Essa alocação realiza-se em função do grau de importância atribuído a cada requisito, no entanto, este modelo como qualquer outro, apresenta suas vantagens e desvantagens conforme elencadas no quadro a seguir.

Modelo Iterativo e Incremental	
Vantagens	**Desvantagens**
Redução dos riscos envolvendo custos a um único incremento. - Se os desenvolvedores precisarem repetir a iteração, a organização perde somente o esforço mal direcionado de uma iteração, não o valor de um produto inteiro.	Dificuldade de gerenciamento. - Isso ocorre porque as fases de do ciclo podem estar ocorrendo de forma simultânea.
Redução do risco de lançar o projeto no mercado fora da data planejada. - Identificando os riscos numa fase inicial o esforço despendido para gerenciá-los ocorre cedo, quando as pessoas estão sob menos pressão do que numa fase final de projeto.	O usuário pode se entusiasmar excessivamente com a primeira versão do sistema e pensar que tal versão já corresponde ao sistema como um todo.
Aceleração do tempo de desenvolvimento do projeto como um todo. - Porque os desenvolvedores trabalham de maneira mais eficiente quando buscam resultados de escopo pequeno e claro.	Como todo modelo esta sujeito a riscos de projeto. - O projeto pode não satisfazer aos requisitos do usuário, a verba do projeto pode acabar. - O sistema de *software* pode ser entregue ao usuário tarde demais.
Reconhecimento de uma realidade frequentemente ignorada. - As necessidades dos usuários e os requisitos correspondentes não podem ser totalmente definidos no início do processo. Eles são tipicamente refinados em sucessivas iterações. Este modelo de operação facilita a adaptação a mudanças de requisitos.	

Quadro 7. Vantagens e Desvantagens do Modelo Iterativo e Incremental.
Fonte: Adaptado de Bezerra, E. (2003).

As vantagens e desvantagens mencionadas relacionam-se a implantação de qualquer modelo, pois os entreves são normais. Claro que obedecendo às particularidades de desenvolvimento de cada um. As expectativas e frustrações ocorridas com o *software*, bem como os riscos no desenvolvimento, cabe aos responsáveis pelo projeto de gerenciá-los, para que não gere desconforto e custos exagerados como também atraso na entrega do projeto.

2.3.5 Metodologia Ágil

A evolução constante no desenvolvimento de *softwares* tornou-se referência quando um grupo de especialistas estabeleceu alguns quesitos. Desde então, aponta-se as metodologias ágeis como uma alternativa às abordagens tradicionais no desenvolvimento de *software*.

Segundo Banki e Tanaka (2008, p. 22), “as metodologias ágeis, como a Extreme Programming (XP) e o Scrum, entre outras, têm despertado atenção crescente do mercado”. Dentre os métodos ágeis, destaca-se a Programação Extrema, ou XP, criada a partir de diversas práticas de sucesso adotadas na indústria e formalizada em 1999 por Kent Beck, com a publicação do livro “Extreme Programming Explained: Embrace Change”. A Programação Extrema propõe um conjunto de valores, princípios e práticas, que visam a garantir o sucesso no desenvolvimento de *software*, em face a requisitos vagos e que mudam constantemente (SATO, 2007, p. 3).

Esse movimento, baseado no ciclo de desenvolvimento incremental e iterativo, foca-se na colaboração do cliente, no valor dos indivíduos e na adaptação às mudanças. Demonstra-se com isso, ganhos de produtividade nos diversos tipos de projetos de desenvolvimento de *software*, eliminando-se documentação excessiva e burocracia, proporcionando interatividade e qualidade no desenvolvimento de *software*.

Em um artigo publicado em 2003, Martin Fowler, cita algumas metodologias apresentadas no quadro abaixo.

Metodologias Ágeis citadas por Martin Fowler	
Metodologias	XP (Extreme Programming)
	Scrum
	Crystal
	Context Driven Testing
	Lean Development
	(Rational) Unified Process
	Agile Modeling
	DSDM (Dynamic Systems Development Method)
	FDD (Feature Driven Development)
	Pragmatic Programming
	Adaptive *Software* Development

Quadro 8. Metodologias Ágeis.
Fonte: Adaptado de Fowler, M. (2003).

Para Highsmith (2002 apud Ludvig e Reinert, 2007, p. 3), "o termo Metodologia Ágil, popularizou-se em fevereiro de 2001, quando um grupo de 17 especialistas criou a Aliança Ágil e estabeleceram o Manifesto Ágil para o desenvolvimento de *software*".

Valores do Manifesto Ágil	
"Manifesto Ágil" *(Agile Manifesto)*	Indivíduos e interações são mais importantes que processos e Ferramentas;
	Software funcionando é mais importante do que documentação completa e detalhada;
	Colaboração com o cliente é mais importante do que negociação de contratos;
	Adaptação a mudanças é mais importante do que seguir o plano inicial.

Quadro 9. Manifesto Ágil.
Fonte: Adaptado de H., J. (2002 apud Ludvig, D.; Reinert J., 2007, p. 3).

Para auxiliar na compreensão do enfoque do desenvolvimento ágil, os membros da Aliança Ágil refinaram as filosofias contidas em seu manifesto.

O Quadro 10 relaciona-se aos doze princípios, aos quais os métodos ágeis de desenvolvimento de *software* devem se adequar.

Doze Princípios do Manifesto Ágil		
Princípios	1	A prioridade é satisfazer ao cliente através de entregas de *software* de valor contínuas e frequentes;
	2	Entregar *software*s em funcionamento com frequência de algumas semanas ou meses, sempre na menor escala de tempo;
	3	Ter o *software* funcionando é a melhor medida de progresso;
	4	Receber bem as mudanças de requisitos, mesmo em uma fase avançada, dando aos clientes vantagens competitivas;
	5	As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente durante todo o projeto;
	6	Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessário para a realização do trabalho;
	7	A maneira mais eficiente da informação circular dentro da equipe é através de uma conversa face a face;
	8	As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;
	9	Atenção contínua a excelência técnica e um bom projeto aumentam a agilidade;
	10	Processos ágeis promovem o desenvolvimento sustentável. Todos envolvidos devem ser capazes de manter um ritmo de desenvolvimento constante;
	11	Simplicidade é essencial;
	12	Em intervalos regulares, a equipe deve refletir sobre como se tornarem mais eficazes e então se ajustar e adaptar seu comportamento.

Quadro 10. Doze Princípios do Manifesto Ágil.
Fonte: Adaptado de C., A. (2000 apud Ludvig, D.; Reinert J., 2007, p. 3-4).

A escolha da metodologia mais adequada para o desenvolvimento de *software* em uma organização, leva-se em consideração os fatores envolvidos, não é um fato ordinário. Por outro lado, trata-se de um fator principal para o sucesso da organização. Embora um processo adequado não possa garantir o sucesso de um projeto, certamente a adoção de um processo inadequado pode comprometê-lo.

3 ENGENHARIA DE REQUISITOS

A engenharia de requisitos trata-se de uma importante parte da engenharia de *software*, onde as empresas investem no aprimoramento técnico, como também o processo onde identifica-se os serviços que o sistema deve oferecer e as restrições que deve respeitar, estabelecendo uma base sólida para o projeto e construção. Sem a engenharia, o resultado do *software* tem a probabilidade de não satisfazer às necessidades dos clientes.

Conforme Aurum e Wohlin (2005 apud Falbo, 2011, p. 1), "engenharia de requisitos é o processo pelo qual os requisitos de um produto de *software* são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de vida do *software*".

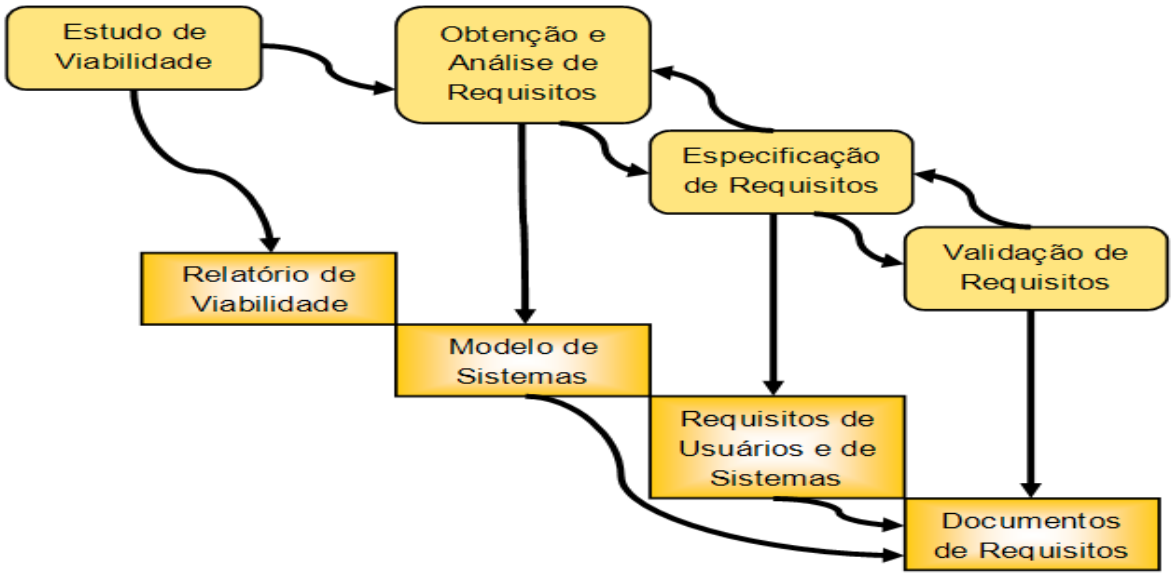


Figura 6. O processo de Engenharia de Requisitos.
Fonte: Adaptado de Sommerville, I. (2003, p. 103).

O processo utilizado na ER varia dependendo do domínio da aplicação, como também das pessoas envolvidas e a organização. Observadas na Figura 6, pode-se elencar as seguintes atividades do processo de engenharia de requisitos.

Processo de Engenharia de Requisitos	
Atividades	Estudo de Viabilidade
	Obtenção e Análise de Requisitos
	Especificação dos Requisitos
	Validação de Requisitos

Quadro 11. Atividades do Processo de Engenharia de Requisitos.
Fonte: Adaptado de K., G.; Sommerville, I. (1998 apud Falbo, R., 2011).

Conforme Quadro 11, as atividades do processo de engenharia de requisitos relacionam-se à condução e a forma iterativa, até que se chegue ao alcance das definições que suficientemente precisas para garantir o sucesso do projeto. Desta forma, o estudo de viabilidade relaciona-se à estimativa de tecnologias e *software*s a um custo e prazo efetivo.

A análise de requisitos fundamenta-se das observações do sistema existente e entrevistas realizadas, para ajudar no compreendimento do esboço do modelo de sistema, e os desenvolvedores a entender o funcionamento do sistema. Já, quanto à especificação dos requisitos refere-se à descrição detalhada e precisa dos requisitos do sistema, proporcionando um embasamento para o contrato entre o cliente e o desenvolvedor.

Por fim, a validação dos requisitos trata-se de uma fase especialmente importante, pois pretende-se demonstrar que o documento de requisitos produzido corresponde, ao sistema que o cliente pretende.

A Engenharia de Requisitos tem se tornado cada vez mais necessária para resolver os problemas encontrados nas organizações com relação à definição de sistemas (GASTALDO; MIDORIKAWA, 2004, p. 1). Desta forma tem-se abordado com a ER o princípio de separação de características por meio de métodos de modelagem, com a possibilidade de reuso, rastreabilidade e evolução, conforme explanação nos tópicos seguintes.

3.1 ANÁLISE DE REQUISITOS

Objetiva-se com a análise de requisitos, o tratamento do processo de definição dos requisitos do *software* a ser desenvolvido. Com isso, elabora-se as atividades criteriosamente, compreendendo o que espera-se do aplicativo. Conforme Oliveira, L. (2009, p. 10), "esta fase visa identificar o tipo de serviço de processamento de dados a ser executado, os objetivos a serrem alcançados, recursos e prazos necessários para a execução do projeto".

A análise de requisitos considera-se em última instância, uma atividade de construção de modelos, onde representa-se o real em uma abstração da realidade em propósito de um sistema, sendo que na análise de requisitos define-se o Diagrama de Contexto, ou o DFD de Contexto, que representa o macro-processo do sistema, e a interação com as entidades que o manipulam/interagem e em outras definições, os requisitos incluem especificações dos serviços que o sistema deve prover e restrições as quais ele deve operar. Neste contexto as propriedades gerais do sistema e restrições devem ser satisfeitas no seu processo de desenvolvimento. Referindo-se ainda à análise de requisitos, encontra-se na literatura requisitos de *software*s separados em requisitos funcionais e requisitos não-funcionais.

Sobre o assunto, Sommerville (2003, p. 82),

> faz uma classificação de requisitos segundo seus níveis de descrição, sendo requisitos de usuário as "declarações, em linguagem natural e também em diagramas sobre as funções que o sistema deve fornecer e as restrições sobre as quais deve operar" e, requisitos de sistemas, como as descrições detalhadas das funções e restrições do sistema.

A partir dessa classificação o autor define os requisitos funcionais e não-funcionais. Onde os requisitos funcionais referem-se às declarações de funções que o sistema deve fornecer e como o sistema deve reagir a entradas específicas, como também deve se

comportar em determinadas situações e os requisitos não-funcionais referem-se às restrições sobre os serviços ou as funções oferecidos pelo sistema.

Requisitos		
Funcionais	**Não-funcionais**	**Domínio**
Compreendem todas as funcionalidades que o sistema deve apresentar considerando aquelas funcionalidades que estão alheias à intervenção ou interação do usuário com o *software*.	São aqueles que se referem ao atendimento de uma demanda não diretamente relacionada às funções do *software*.	São requisitos que originam do domínio de aplicação do sistema e que refletem características desse domínio, podendo ser funcionais ou não funcionais.

Quadro 12. Classificação dos Requisitos.
Fonte: Adaptado de Silva, C.; Souza, K.; Dantas, S., (2006, p. 65).

Entende-se que os requisitos funcionais dependem do tipo de *software* que está sendo desenvolvido e dos usuários de *software* que se espera atingir. Já os Requisitos não-funcionais referem-se ao padrão estético das interfaces gráficas do *software*, o tempo de resposta de determinada ou tipo de linguagem de programação a ser utilizado para o desenvolvimento.

Na análise de requisitos os modelos são fundamentais no desenvolvimento de sistemas. Segundo Berwanger (2009, p. 13), os modelos são tipicamente construídos para:

a) Possibilitar o estudo do comportamento do sistema;
b) Facilitar a comunicação entre os componentes da equipe de desenvolvimento, clientes e usuários;
c) Possibilitar a discussão de correções e modificações com o usuário;
d) Formar a documentação do sistema.

Para cada modelo, enfatiza-se um conjunto de características, da realidade, possuindo assim, níveis de abstração. Relacionando-se ao desenvolvimento de sistemas, consideram-se três níveis.

Níveis de Abstração		
Conceitual	**Lógico**	**Físico**
Considera características do sistema independentes do ambiente computacional (*hardware* e *software*), no qual o sistema será implementado. Essas características são dependentes unicamente das necessidades do usuário. Modelos conceituais são construídos na atividade de análise de requisitos.	Características dependentes de um determinado tipo de sistema computacional. Essas características são, contudo, independentes de produtos específicos. Tais modelos são típicos da fase de projeto.	Características dependentes de um sistema computacional específico, isto é, uma linguagem e um compilador específico, um sistema gerenciador de bancos de dados específico,o *hardware* de um determinado fabricante etc.

Quadro 13. Níveis de Abstração.
Fonte: Adaptado de Berwanger (2009, p. 14).

Conforme elencados acima, nas primeiras etapas do processo de desenvolvimento, o engenheiro de *software* representa o sistema pelos conceituais. Posteriormente, as características lógicas e físicas são representadas em novos modelos.

3.2 MODELAGEM LÓGICA

Inicia-se o modelo lógico a partir do modelo conceitual, considerando as abordagens, Relacional, Hierárquica e Rede. Levando em conta, limitações e implementação de recursos e nomenclatura. Descreve-se o modelo lógico por meio das estruturas que estarão contidas no banco de dados, não considerando ainda nenhuma característica específica de SGBD, resultando em um esquema lógico de dados. Um modelo lógico trata-se de uma definição de um banco de dados no grau de abstração visto pelo usuário do SGBD. Assim, o modelo lógico torna-se dependente do tipo particular de SGBD que está sendo usado (HEUSER, 2001, p. 17).

Apresenta-se na figura um modelo lógico para o BD o qual deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas.

TipoDeProduto

CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Produto

CodProd	DescrProd	PreçoProd	CodTipoProd
1	PC descktop modelo x	2.500	1
2	PC notebook ABC	3.500	1
3	Imp. jato de tinta	600	2
4	Impressora laser	800	2

Figura 7. Modelo Lógico para Banco de Dados.
Fonte: Adaptado de Heuser, C. (2001).

Objetiva-se com a modelagem lógica, transformar o modelo conceitual obtido na primeira etapa em um modelo lógico, o qual define como o banco de dados será implementado em um SGBD específico.

A modelagem refere-se ao processo de construção de um modelo, o qual consiste em uma abstração de algo real, podendo ser feito antes mesmo deste algo se consolidar. Segundo Berwanger (2009, p. 14), “um modelo enfatiza um conjunto de características da realidade, que corresponde à dimensão do modelo”. Dá-se com o modelo uma visão do projeto antes de sua execução, sendo ela, externa do projeto, ajudando a reconhecer e até a corrigir erros antes mesmo desses acontecerem, tornando-se um projeto confiável. Constroem-se modelos para compreender o sistema que estamos desenvolvendo.

3.3 MODELAGENS DE SISTEMAS

A modelagem de sistemas inicia-se por uma distinção que associa a maneira de perceber os fenômenos ao modo de pensamento que predomina em cada indivíduo, na observação de um determinado fenômeno, num momento específico do tempo. Defini-se então o Sistema, como uma coletânea de estruturas e recursos que são interagidos segundo uma lógica de tal forma a alcançar um ou mais objetivos. Para Silva, (2005), em seu artigo

publicado na internet, “os estudos destes sistemas podem dar-se sob diferentes formas de abordagem”.

Formas de Abordagem		
Formas	**Abordagem**	**Resultado**
1ª	Refere-se na intervenção direta sob rotinas operacionais promovendo implementações e, ou, alterações de procedimentos até que sejam obtidas as condições ideais.	Estas ações fazem requerer do tomador de decisão a condução de estudos preliminares e experiência, para que as alterações não minorem a performance do sistema.
2ª	Refere-se a utilização de modelos que representem os sistemas reais. Os modelos podem apresentar-se como protótipos ou como modelos matemáticos.	Este presta-se à soluções analíticas, como por exemplo um modelo de regressão, ou a simulação, permitindo assim, reconstituir a rotina funcional de um dado sistema real.

Quadro 14. Formas de Abordagem.
Fonte: Adaptado de Silva, (2005).

Um sistema de *software* bem sucedido refere-se aquele, capaz de atender às necessidades dos usuários com qualidade. Para produzir tal sistema de *software*, de maneira eficiente, com eficácia de recursos e de maneira previsível, utiliza-se a modelagem de sistemas. A Figura 8 exemplifica o processo de modelagem de sistemas.

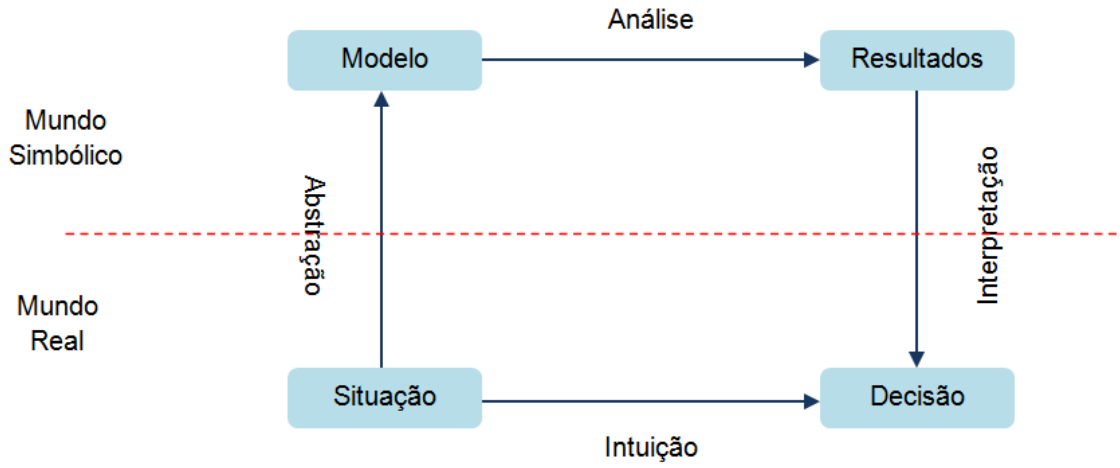


Figura 8. Processo de Modelagem de Sistemas.
Fonte: Adaptado de Gomide, F., (2007).

Observa-se na Figura 8, que a abstração e a interpretação são cruciais para o processo de modelagem para a tomada de decisão. Neste contexto a modelagem de *software* consiste em auxiliar na organização das informações, estabelecendo um escopo para o projeto que descreva a real utilidade perante o *software*, levando em consideração as análises de requisitos o *software* terá sua validação em seu conjunto.

3.4 UML – LINGUAGEM DE MODELAGEM UNIFICADA

A UML, Linguagem de Modelagem Unificada (Unified Modeling Language), surgiu para oferecer uma modelagem de forma a abranger todas as visões de um sistema de *software*, possibilitando uma maior interação entre analista, projetistas, programadores e demais profissionais envolvidos no desenvolvimento de um sistema. UML ou Unified Modeling Language é uma unificação dos métodos OMT, Booch e OOSE que está sendo submetido a OMG para a padronização (MAZZOLA, 2009, p. 9.5).

A UML não equivale a um modelo de processo de *software* ou uma metodologia de desenvolvimento de sistemas, a UML trata-se de uma notação, um mecanismo para apontar o problema de forma a expor a essência do domínio de um aplicativo. Combinando-se a UML com um modelo de processo resulta na criação de aplicativos bem-sucedidos.

Enfatiza-se que a UML consiste em ser somente uma linguagem e, portanto, trata-se apenas de parte de um método de desenvolvimento de *software*, independente do processo de *software* a ser usado, ainda que seja mais adequada a processos de desenvolvimento orientados a objetos. Com o objetivo de descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos. Utiliza-se para criar modelos de sistemas de *software*, mas utiliza-se também para representar sistemas mecânicos sem nenhum *software*.

Conforme Berwanger (2009, p. 16), “é uma linguagem padrão para a elaboração da estrutura de projetos de *software*”. A UML fornece uma maneira para capturar e discutir exigências (Diagramas de Caso do Uso). Encontra-se diagramas para capturar as partes do sistema de *software* que realizam determinadas exigências (Diagramas de Colaboração). Diagramas para capturar exatamente como aquelas partes do sistema realizam suas exigências (Diagramas de Sequência e de Gráfico de Estado). Finalmente, diagramas para mostrar como tudo se unem para juntos serem executados (Diagramas de Componentes e de Distribuição). Encontram-se diversos tipos, alguns com finalidades específicas e alguns com usos mais genéricos.

3.5 DIAGRAMAS

Entende-se por diagramas o estudo gráfico dos esforços que retratam os valores dos esforços simples ao longo da estrutura, permitindo a visualização das variações desses esforços de uma seção para outra. Neste sentido compreende-se como sendo uma representação visual estruturada e simplificada de um determinado conceito, onde em quase todas as áreas do conhecimento humano, existem diversos tipos de diagramas.

Dentro do estudo em questão, estuda-se com a utilização dos diagramas de UML, que de e acordo com Ferrari (2006, p. 47), “a UML modela um sistema e sua interação com o usuário por meio de diagramas padronizados”.

Os diagramas UML esboçam todas as visões de um projeto de sistema de *software*. São treze diagramas apresentados na literatura, no entanto a abordagem deste estudo enfatizará somente os diagramas de caso de uso (Use Case), diagrama de sequência, diagrama de colaboração e diagrama de atividades.

3.5.1 Diagrama de Caso de Uso

Um Diagrama de Caso de Uso modela a funcionalidade fornecida pelo sistema. Segundo Savi (2009, p. 84), diagrama de caso de uso mostra como o sistema vai interagir com os usuários (pessoas ou outros sistemas). Desta forma diagramas de caso de uso descrevem as interações e funcionalidades entre as categorias de usuários e o sistema.

Para Pressman (2006, p. 153) “descrevem uma sequência de ações que são realizadas por um ator à medida que o ator interage com o *software*”.

Compreende-se que a finalidade principal deste diagrama concede em ajudar as equipes de desenvolvimento a visualizar as exigências funcionais de um sistema. Neste diagrama, os atores relacionam-se com os processos essenciais, assim como os relacionamentos entre casos diferentes de uso. Este diagrama nos mostra grupos de casos de uso, onde os seres humanos interagem com o sistema, ou outros sistemas interagem com o mesmo. Observa-se abaixo um exemplo de diagrama de casos de uso.

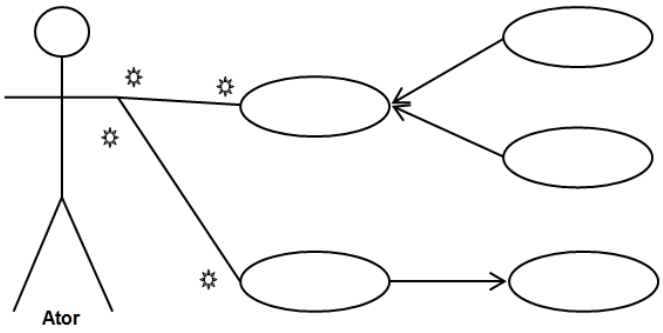


Figura 9. Exemplo de Diagrama de Caso de Uso.
Fonte: Costa et al., (2001 apud Junqueira, F., 2006, p. 99).

Utilizam-se casos de uso, para estabelecer a funcionalidade desejada do sistema e realizar testes para a comunicação dos participantes, e por fim, a validação do sistema. Em um projeto, cada participante tem a sua visão e para o sucesso, todos devem interagir falando a

mesma linguagem, focando o mesmo objetivo. A UML com o diagrama de caso de uso vem tentar exatamente isso, dar uma visão funcional do sistema como um todo e todas as interações possíveis, trazendo benefícios conforme apresentado no Quadro 15.

Benefícios conforme Leffingwell	Os Casos de Uso são relativamente fáceis de escrever e mais fáceis de ler;
	Forçam os desenvolvedores a pensar no projeto de um sistema da perspectiva de um usuário;
	Dão contexto para os requisitos do sistema. Pode-se compreender porque uma exigência existe e como o sistema chega aos seus objetivos;
	É uma ótima ferramenta no processo da análise, ajudando a compreender o que o sistema necessita fazer e como realizar isso;
	É uma ferramenta que atua no projeto e no desenvolvimento. Reduz o risco de erros ao se implementar os requisitos do sistema, por dar uma visão geral de todos os requisitos e onde cada um deve chegar;
	É útil também no processo de teste, ajudando a definir se o sistema chegou realmente onde deveria chegar.

Quadro 15. Benefícios de Casos de Uso.
Fonte: Adaptado de Leffingwell, (2003 apud Ferrari, T., 2006).

A descrição dos casos de uso provê informações que ajudam ao se especificar as propriedades das classes necessárias para atuar como um caso de uso, representando as funcionalidades de alto nível do sistema, detalhando o que o sistema deveria fazer.

3.5.2 Diagrama de Sequência

Um Diagrama de Sequência modela a sequência funcional do sistema de *software*, dando uma sequência temporal aos itens presentes nele. Estes diagramas fornecem uma representação abreviada da maneira pelas quais as ações de usuário colaboram com elementos estruturais do sistema, podendo-se perceber a sequência de mensagens enviadas entre objetos, mostrando interações entre componentes do sistema.

Segundo Pender (2003 apud Ferrari, 2006, p. 57), os diagramas de sequência são valiosos por que:

a) Tem um foco estreito que ajuda a visualizar comandos e dados que estão sendo utilizados durante a execução de uma tarefa específica.
b) Identifica explicitamente a comunicação requerida para cumprir uma interação. Isto ajuda a validar ou descobrir as relações requeridas por uma classe.
c) Identifica explicitamente os objetos que participam em uma interação. Isto ajuda validar ou descobrir as características de uma classe.
d) Identifica explicitamente os dados que são passados entre as partes das interações. Os dados têm que pertencer a uma classe em algum lugar no projeto.

Utiliza-se um diagrama de sequência geralmente para definir rigorosamente a lógica para um cenário de caso de uso. Tradicionalmente os diagramas de sequência mostram os tipos de objetos envolvidos no caso de uso, as mensagens que trocam entre si e qualquer valor retornado associado com a mensagem (JUNQUEIRA, 2006, p. 100).

Revisa-se com os diagramas de sequência, o trabalho realizado forçando o projetista a rever os passos por meio de lógica para preencher os cenários de caso de uso.

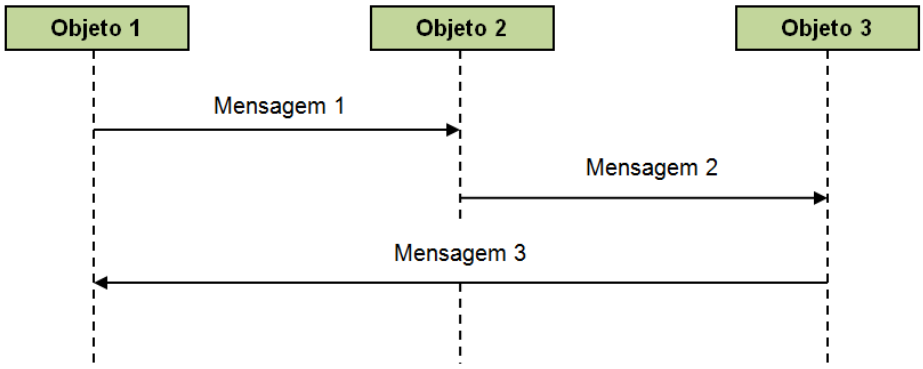


Figura 10. Exemplo de Diagrama de Sequência.
Fonte: Adaptado de Costa et al., (2001 apud Junqueira, F., 2006, p. 100).

Representam-se por meio destes diagramas, os aspectos comportamentais dos objetos, mostrando que métodos são necessários para satisfazer um caso de uso específico. Conforme observado na Figura 10, Booch (2000 apud Ferrari, 2006, p. 57), relatam que “o Diagrama de Sequência consiste em um número de objetos mostrados em linhas verticais”.

O decorrer do tempo é visualizado observando-se no sentido vertical, de cima para baixo, as mensagens enviadas por cada objeto simbolizam-se por setas entre os objetos que se relacionam.

3.5.3 Diagrama de Colaboração

Com o diagrama de colaboração consegue-se verificar a interação dinâmica entre objetos, a relação entre um conjunto de objetos e a troca de mensagens existentes entre eles. Segundo Savi (2009, p. 84), "o diagrama de colaboração mostra as interações entre vários componentes do sistema". Neste contexto, pode-se utilizar o diagrama de colaboração para substituir o diagrama de sequência.

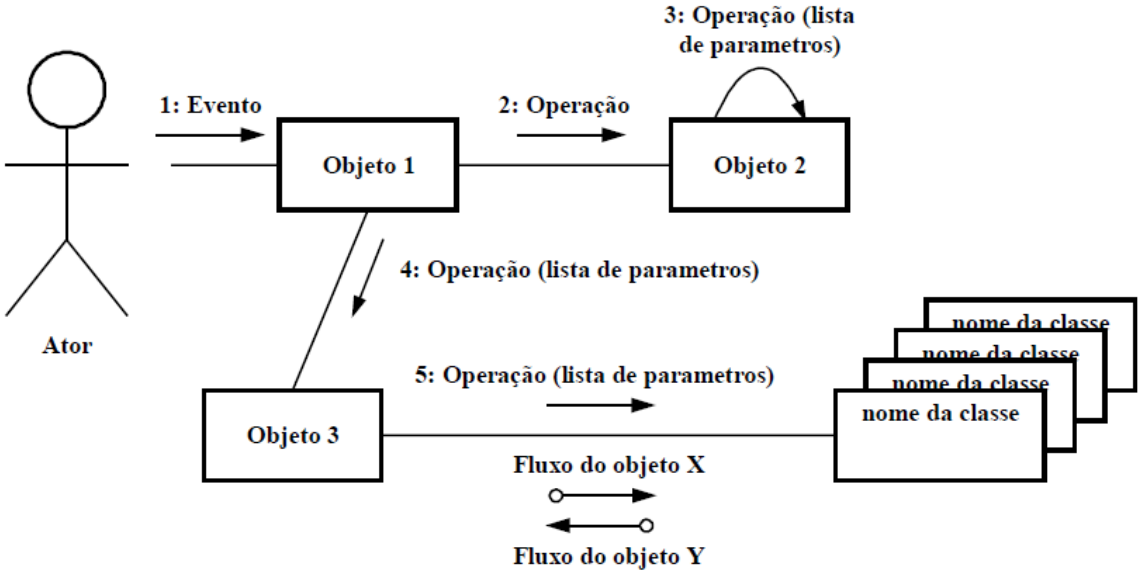


Figura 11. Exemplo de Diagrama de Colaboração.
Fonte: Douglass (1998 apud Junqueira, F., 2006, p. 101).

Verificou-se na Figura 11 que o diagrama de colaboração, também obedece a uma sequência. Neste sentido Pender (2003 apud Ferrari 2006, p. 58), relata que "o diagrama de colaboração é quase o mesmo que o diagrama de sequência e que a diferença é a perspectiva".

Observa-se que ambos os diagramas modelam interações entre objetos para uma tarefa específica, mas enquanto o diagrama de sequência enfatiza o arranjo na sequência de interações sobre o tempo, o diagrama de colaboração enfatiza a organização estrutural dos objetos que enviam e recebem mensagem, desta forma, modela a forma das interações utilizam a estrutura dos objetos e a participação em seus relacionamentos.

3.5.2 Diagrama de Atividades

Representam-se por meio do diagrama de atividades, as ações (atividades) dos objetos e seus resultados. Trata-se de uma variação do diagrama de estados, porém representa as ações e os resultados que ocorrem em determinada mudança de estado de um objeto. Os diagramas de atividades são usados para detalhar classes, implementação de operações e casos de uso, enquanto os diagramas de estado são usados para especificar o comportamento global de um tipo.

Segundo Savi (2009, p. 84), "o diagrama de atividade mostra como um subsistema ou um objeto realiza uma operação, também conhecido como diagrama de fluxo de dados". Documentam-se os diagramas de atividades com uma breve descrição da atividade e uma indicação da ação que ocorre (ou das ações que ocorrem) durante um processo.

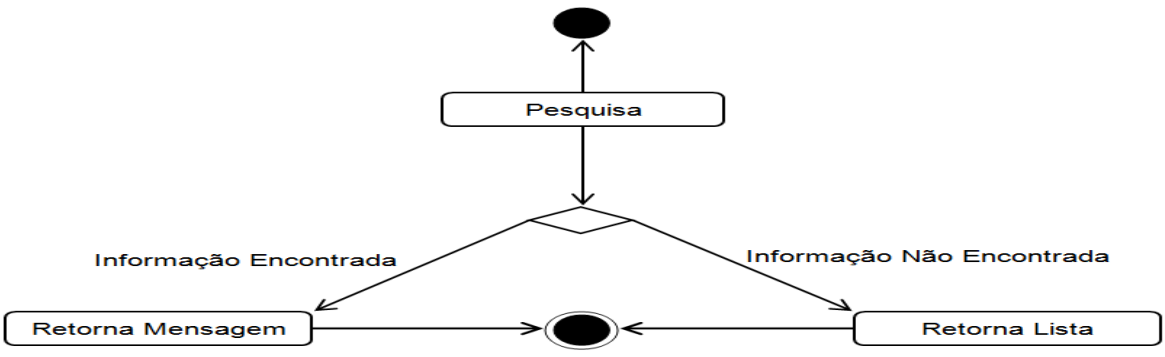


Figura 12. Exemplo de Diagrama de Atividades.
Fonte: Camargo, L. (2009, p. 229).

Nesse exemplo da Figura 12, verifica-se que a atividade representada consiste em uma pesquisa de informação, isso geralmente é feito em uma ferramenta de busca, que ao encontrar a informação pesquisada retorna uma lista de resultados e ao não encontrar tal informação retorna uma mensagem que não foi encontrado nenhum resultado.

Conforme Junqueira (2006, p. 101), “o diagrama de atividades modela a sequência de atividades em um processo, bem como seus elementos ativos e passivos”. O diagrama de atividades mostra um conjunto de atividades, o fluxo de cada atividade para com outra, e os objetos que realizam ou sofrem ações.

4 BANCO DE DADOS

Os sistemas de gerenciamento de banco de dados (SGBD) surgiram no início da década de 70 com o objetivo de facilitar a programação de aplicações de banco de dados. Conforme Alexandruk (2011, p. 2), “um banco de dados pode ser definido como uma coleção de dados integrados, que tem por objetivo atender a uma comunidade de usuários”. Como também um conjunto de dados persistentes e manipuláveis que obedecem a um padrão de armazenamento. Por “dados” pode-se compreender como fatos conhecidos que podem ser armazenados e que possuem um significado implícito.

4.1 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

Ao longo do tempo, os SGBD’s evoluíram desses sistemas de arquivos de armazenamento em disco, criando-se novas estruturas de dados com o objetivo de armazenar informações. Com esta evolução os SGBD’s passaram a utilizar diferentes formas de representação, ou modelos, para descrever a estrutura das informações contidas em seus bancos de dados.

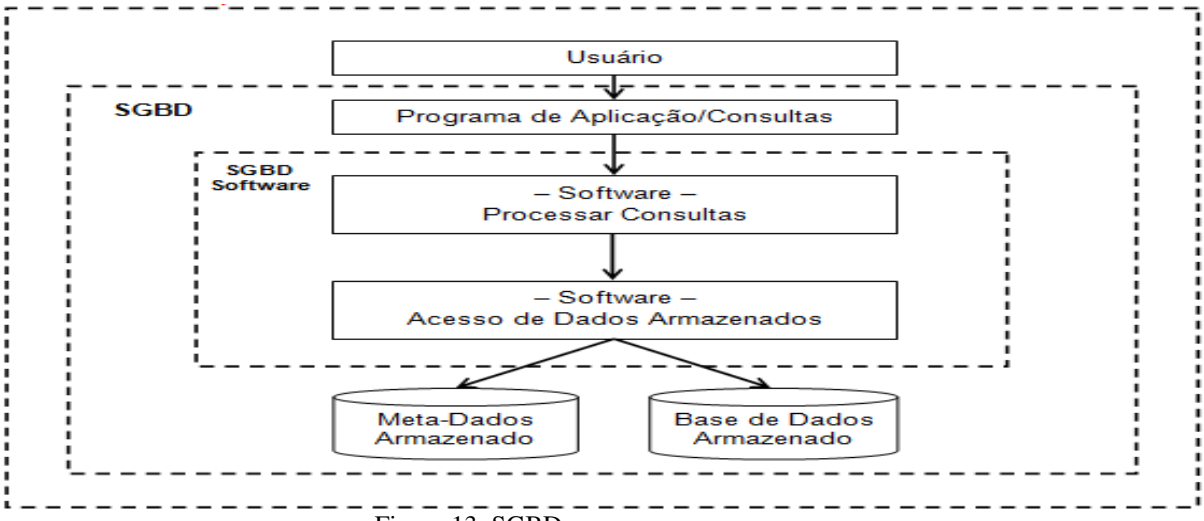


Figura 13. SGBD.

Segundo Alexandruk (2011, p. 2), “o Sistema de Gerenciamento de Banco de Dados é o *software* que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados”. Normalmente, os programas não contêm todo código referente à exibição dos dados na interface e utilizam-se de gerenciadores de interface de usuário e conjuntos de rotinas que incluem as funcionalidades que um programador vai necessitar frequentemente, ao construir uma interface, neste ponto a maioria dos programas comunica-se com os usuários por meio de interfaces gráficas de janelas.

Alexandruk (2011, p. 2) enfatiza que os Bancos de dados informatizados apresentam as seguintes vantagens, conforme mencionados no Quadro 16.

Banco de Dados Informatizados	Vantagens
	Compacto (elimina arquivos de papéis);
	Rápido;
	Integrado (vários aplicativos utilizam o mesmo repositório de dados);
	Compartilhado (vários usuários podem acessar);
	Seguro (controle de acesso);
	Padronizado;
	Consistente;
	Suporte a transações.

Quadro 16. Vantagens do Banco de Dados Informatizados.
Fonte: Adaptado Alexandruk, M. (2011).

Permite-se com um SGBD, que os usuários criem e manipulem bancos de dados de propósito gerais. Desta forma, o conjunto formado por um banco de dados, mais as aplicações que manipulam o mesmo é chamado de “Sistema de Banco de Dados”.

4.2 ARQUITETURA E MODELAGEM DE BANCO DE DADOS

As primeiras arquiteturas usavam mainframes para executar o processamento principal, bem como o de todas as funções do sistema, incluindo os programas aplicativos de interface com o usuário, como também a funcionalidade do SGBD. Com esta arquitetura, os usuários acessavam somente pelos terminais, os quais não possuíam processamento e apenas

permitia visualização. Desta forma, os processamentos de todos os dados eram feitos remotamente, conectados por redes de comunicação.

Atualmente, devem-se considerar alguns aspectos relevantes para atingir a eficiência e a eficácia dos sistemas informatizados desenvolvidos, a fim de atender seus usuários nos mais variados domínios de aplicação.

Conforme Takai; Italiano e Ferreira (2005, p. 9-10), enumeram os seguintes aspectos mais relevantes:

> a) Os projetos Lógico e Funcional do Banco de Dados devem ser capazes de prever o volume de informações armazenadas a curto, médio e longo prazo. Os projetos devem ter uma grande capacidade de adaptação para os três casos mencionados;
> b) Deve-se ter generalidade e alto grau de abstração de dados, possibilitando confiabilidade e eficiência no armazenamento dos dados e permitindo a utilização de diferentes tipos de gerenciadores de dados através de linguagens de consultas padronizadas;
> c) Projeto de uma interface ágil e com uma "rampa ascendente" para propiciar aprendizado suave ao usuário, no intuito de minimizar o esforço cognitvo;
> d) Implementação de um projeto de interface compatível com múltiplas plataformas (UNIX, Windows NT, Windows Workgroup, etc);
> e) Independência de Implementação da Interface em relação aos SGBDs que darão condições às operações de armazenamento de informações (ORACLE, SYSBASE, INFORMIX, PADRÃO XBASE, etc).
> f) Conversão e mapeamento da diferença semântica entre os paradigmas utilizados no desenvolvimento de interfaces (Imperativo (ou procedural), Orientado a Objeto, Orientado a evento), servidores de dados (Relacional) e programação dos aplicativos (Imperativo, Orientado a Objetos).

Encontra-se atualmente, várias tendências para arquitetura de Banco de Dados, nas mais diversas direções e modelos.

De acordo com Alexandruk (2011, p. 2), “o modelo de dados refere-se à descrição formal da estrutura de um banco de dados”. Desta forma, para construir um modelo de dados, utiliza-se uma linguagem de modelagem de dados. Estas linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens textuais ou linguagens gráficas.

O quadro a seguir demonstra os objetivos da modelagem de dados.

Objetivo da Modelagem de Dados
- Representar o ambiente observado - Documentar e normalizar - Fornecer processos de validação - Observar processos de relacionamentos entre objetos

Quadro 17. Objetivos da Modelagem de Dados.
Fonte: Adaptado de Macoratti, (http://www.macoratti.net/cbmd1.htm).

Observa-se no projeto de banco de dados, normalmente considera-se três níveis de abstração de modelo de dados conforme elencado abaixo.

	Modelos	Conceitos	Características
Projeto de Banco de Dados	Modelo Conceitual	Representa as regras de negócio sem limitações tecnológicas ou de implementação por isto é a etapa mais adequada para o envolvimento do usuário que não precisa ter conhecimentos técnicos.	- Visão Geral do negócio - Facilitação do entendimento entre usuários e desenvolvedores - Possui somente as entidades e atributos principais - Pode conter relacionamentos n para m.
	Modelo Lógico	Leva-se em conta limites impostos por algum tipo de tecnologia de banco de dados. (banco de dados hierárquico, banco de dados relacional, etc.).	- Deriva do modelo conceitual e via a representação do negócio - Possui entidades associativas em lugar de relacionamentos n:m - Define as chaves primárias das entidades - Normalização até a 3ª. forma normal - Adequação ao padrão de nomenclatura - Entidades e atributos documentados
	Modelo Físico	Leva em consideração limites impostos pelo SGBD (Sistema Gerenciador de Banco de dados) e pelos requisitos não funcionais dos programas que acessam os dados.	- Elaborado a partir do modelo lógico - Pode variar segundo o SGBD - Pode ter tabelas físicas (log, líder, etc.) - Pode ter colunas físicas (replicação)

Quadro 18. Modelagem de Dados.
Fonte: Adaptado de Macoratti, (http://www.macoratti.net/cbmd1.htm).

Observou-se que a modelagem de dados trata-se de uma técnica usada para a especificação das regras de negócios e as estruturas de dados de um banco de dados, fazendo parte do ciclo de desenvolvimento de um sistema de informação e torna-se vital para o resultado do projeto. A modelagem de dados consiste no desenho do sistema de informações, concentrando-se nas entidades lógicas e nas dependências lógicas entre essas entidades.

4.3 MODELO DE BANCO DE DADOS

Os bancos de dados apareceram no final dos anos 60, numa época em que a necessidade de um sistema de gestão da informação flexível passava a ser necessária diante da evolução constante, no entanto já no fim dos anos 90, as bases relacionais relacionavam-se aos bancos de dados mais comuns.

De acordo com Laundon & Laundon (1999 apud Araújo, 2003, p. 8), “existem três modelos de projeto de lógico de banco de dados: o modelo hierárquico, o modelo em rede e o modelo relacional”. Os modelos citados pelo autor, tratam-se, de formas de como os dados são organizados em um banco de dados e que dependem da natureza de cada problema que se deseja solucionar.

Modelo Hierárquico

O modelo hierárquico trata-se do primeiro modelo de SGBD, onde classifica-se os dados de acordo em forma hierárquica descendente, utilizando-se de apontadores entre os diferentes registros.

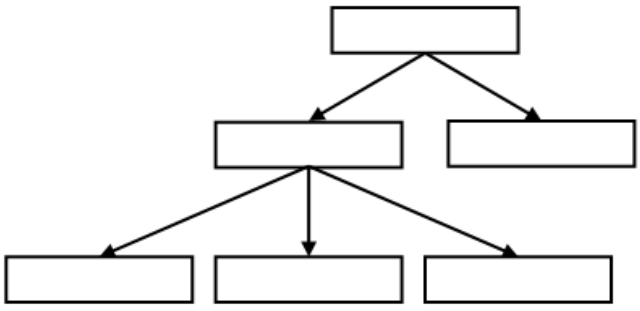


Figura 14. Modelo Hierárquico.

Observa-se que o modelo de Banco de Dados Hierárquico organiza os dados de cima para baixo, de maneira organizada e de forma hierárquica. Conforme Langa (2007),

“este modelo utiliza árvores para a representação lógica dos dados, composta de elementos chamados nós, onde o nível mais alto da árvore denomina-se raiz”. De acordo com a menção deste autor, a cada elemento chamado nó, representa um registro com seus correspondentes campos. Compreende-se que essa estrutura hierárquica funciona bem com alguns bancos de dados, tornando-se difícil quando não existe uma hierarquia natural entre os tipos de registro.

Modelo em Rede

Semelhante ao o modelo hierárquico, o modelo em rede utiliza-se apontadores para os registros. Desta forma a estrutura não necessariamente utiliza-se do formato em árvore invertida e descendente.

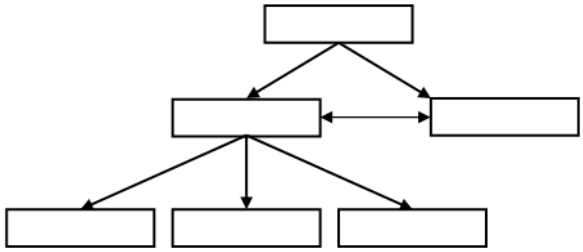


Figura 15. Modelo em Rede.

Neste modelo as entidades se representam como nós e suas relações são as linhas que os unem, desta forma nesta estrutura qualquer componente pode se relacionar com qualquer outro (LANGA, 2007). Comparando-se ao modelo hierárquico, neste modelo, um filho pode ter vários pais sendo que no modelo em rede, o tipo de registro representa um nó.

Estes tipos de sistemas proporcionam capacidades mais complexas de estruturas de dados do que os sistemas hierárquicos, tornando-se de difícil adaptação a mudanças ou alterações de estruturas no Banco de Dados.

Modelo Relacional

No modelo relacional, registram-se os dados em duas dimensões, isto é, linhas e colunas.

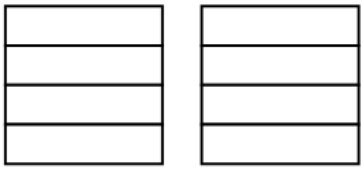


Figura 16. Modelo Relacional.

Para Araújo (2003, p. 9), "o modelo de Banco de Dados Relacional é a mais recente estrutura de banco de dados. Representa todos os dados do banco de dados em tabelas simples bidimensionais denominadas relações". Observa-se que este modelo trata-se do mais utilizado e emprega o uso de tabelas bidimensionais na representação lógica dos dados e suas relações.

O elemento principal deste modelo trata-se da relação que se representa mediante uma tabela e segundo Langa (2007), possuem características principais conforme elencadas no Quadro 19.

Características do Modelo de Banco de Dados Relacional	
Banco de Dados Relacional	Pode ser entendido e usado por qualquer usuário;
	Permite ampliar o esquema conceitual sem modificar as aplicações de gerenciamento;
	Os usuários não necessitam saber onde se encontram os dados fisicamente.

Quadro 19. Características do Modelo Relacional.
Fonte: Adaptado de Langa, S. (2007).

O modelo relacional foi definido por Edgar Frank "Ted" Codd em 1970. O modelo relacional apresenta dois aspectos fundamentais: uma álgebra permitindo manipular tabelas e um mecanismo de concepção de relações (DIAS, 2002, p. 2).

A criação do Banco de Dados Modelo Relacional deu-se, ao fato da limitação dos dois outros modelos, surgindo com a intenção de superar as limitações dos antigos modelos.

O modelo relacional trata-se do mais moderno, bem como possui a mais adequada metodologia, o que o deixa em situação privilegiada sobre os demais, no desenvolvimento de projeto lógico de banco de dados.

4.4 SISTEMA DE BANCO DE DADOS ORIENTADOS A OBJETOS

Segundo Silveira et al (2006, p. 2), “os sistemas de banco de dados orientado a objetos, cuja sigla em inglês é OODBMS, surgiram na década de 80, fruto de pesquisas sobre o armazenamento de dados estruturados sob a forma de grafos”.

Manipula-se os objetos de um OODBMS, por uma linguagem de programação como C++ ou Java, resultando em uma boa interação entre a linguagem de programação e o banco de dados. Observa-se no modelo relacional, uma fase clara de construção de tabelas em SQL, já no modelo orientado a objetos, define-se de maneira explicita e implícita. Desta forma, no modo implícito as classes não são definidas e persistindo-se um objeto a classe acompanhará. No modelo explícito, especifica-se a interface no banco de dados.

Conforme Silveira, et al (2006, p. 7), “o SGBD Objeto-Relacional (SGBDOR) emprega um modelo que coloca a orientação a objetos em tabelas, unindo dois paradigmas em um só”. Essa tecnologia permite que as aplicações persistam os dados de forma transparente, em qualquer tipo de banco. Para a aplicação, trata-se como se existisse um banco de dados orientado a objeto servindo de repositório.

5 *SOFTWARES* OBJETO DE ESTUDO

O estudo constante em busca de alternativas, técnicas e ferramentas de desenvolvimento de produtos de *softwares* que satisfaçam as necessidades dos diversos ramos de negócios, tornou-se fundamental para as empresas fornecedoras destes produtos e desenvolvedores dotados deste conhecimento.

Segundo Queiroz, (2001, p. 83), "anteriormente as empresas de *software* estavam inseridas em um mercado dinâmico, marcado por uma constante evolução das tecnologias, técnicas e métodos de desenvolvimento". Com o surgimento de produtos e serviços concorrentes, por pressão, por qualidade e rapidez fizeram com que evoluíssem para atender a demanda e as necessidades dos clientes.

Neste sentido, apresenta-se uma abordagem dos setores de saúde, industrial e comercial, objetos de estudo no desenvolvimento do protótipo de modelagem lógica de sistemas.

Saúde

Os *softwares* para área de saúde, permitem aos usuários do sistema terem acesso às informações básicas dos clientes como também as informações dos prontuários.

Para Oliveira, C.; Barros e Oliveira (2010, p. 1), "a utilização das técnicas de Engenharia de *Software* e Usabilidade permite a construção de *softwares* funcionais, com qualidade, especialmente quando o usuário é envolvido durante o processo".

Ressalta-se a importância dos *softwares* da área da saúde, da confiabilidade e eficiência que deve apresentar, pois a consistência das informações ali processadas e armazenadas são fundamentais, para o profissional e o paciente.

Industrial

Preocupam-se, os administradores cada vez mais em tempo real, ter as informações das empresas, neste contexto a busca por *softwares* mais completos que atendam as necessidades destes clientes tem aumentado ao longo dos anos. Desta forma, programas que apresentam os números reais dos estoques, e demais informações de controles e cadastros, em relatórios tem despertado interesse neste mercado.

Comercial

O *Software* para utilização comercial foi desenvolvido para o gerenciamento completo de comércios auxiliando no controle de clientes, fornecedores, transportadoras, funcionários, bem como os aplicativos de estoques, contas a receber e todas as ferramentas que auxiliam a gestão do negócio.

Relata Branco (2010) em um site de manutenção e suprimentos, que “com a automatização comercial, os erros e falhas praticamente são inexistentes, e as tarefas, antes complexas, tornaram-se fáceis, rápidas, práticas e altamente eficientes”.

Neste sentido os benefícios da utilização de um sistema apropriado, serão percebidos pelos clientes, parceiros e fornecedores, tornando-se mais competitiva no mercado.

6 FERRAMENTAS UTILIZADAS

Encontra-se, diversas ferramentas utilizadas por desenvolvedores de sistemas, no entanto cada uma permite a utilização de acordo com a necessidade ou gosto do programador. Neste contexto, elenca-se as ferramentas a serem utilizadas no desenvolvimento deste estudo, sendo elas escolhidas para que os objetivos sejam alcançados.

Astah Community

A ferramenta Astah Community trata-se de uma boa ferramenta de modelagem UML gratuita. Conforme Souza, L., (2009) em seu artigo na internet "a ferramenta Astah Community possui a versão Community, com limitações e Professional". Como toda versão free, a Astah Community possui limitações de recursos, mas que atende as necessidades do dia a dia. O Quadro 20 apresenta alguns dos recursos da ferramenta Astah Community.

Astah Community	
Recursos	Suporte a UML 2.1
	Diagramas de Classe, Caso de Uso, Sequência, Atividade, Comunicação, Máquina de Estado, Componentes, Implantação, Estrutura de Composição, Objetos e Pacotes.
	Ajustes de alinhamento e tamanho dos diagramas
	Impressão dos diagramas (com a marca d'água da ferramenta)
	Exportação das imagens dos diagramas (com a marca d'água da ferramenta)

Quadro 20. Características do Modelo Relacional.
Fonte: Adaptado de Souza, L. (2009).

SQL Server Express

Utilizado como banco de dados o SQL Server Express foi desenvolvido tendo em mente duas utilizações básicas. A primeira, como um servidor de produtos, especialmente

como um Web Server ou Database Server. O segundo, como um cliente stand-alone, onde a aplicação não precise depender de uma rede para obter acesso aos dados (PINHEIRO, 2006).

Visual FoxPro

O visual FoxPro trata-se de uma ferramenta de manipulação de dados e feito exclusivamente para “Dados”. Segundo Heringer (2004), “o Visual FoxPro, como ferramenta de desenvolvimento de aplicações em ambiente visual, possui todas as características, e mais algumas, das principais ferramentas do mesmo nível e em uso pelo mercado”.

O Visual FoxPro destina-se à orientação a objetos, acesso a bancos de dados desenvolvimento de interface gráfica, suporte a XML, desenvolvimento rápido entre outras características. Encontra-se as características do Visual FoxPro mencionadas em livros e internet, o que permite a discussão mais ampla dos recursos que podem ser obtidos.

7 TRABALHOS CORRELATOS

Atualmente vimos esforços dos profissionais da área de engenharia de *software* para contribuições referentes a melhorias e soluções de problemas ligados a desenvolvimento e implantação. Podem-se perceber inúmeros artigos e monografia que auxiliam estas e contribuem para esta boa prática. A correlação dos trabalhos citados a seguir, se dá na condição de pesquisa no desenvolvimento de *softwares* e alternativas para atenderem as necessidades dos clientes. A criação de protótipos e modelagem de dados é o ponto base para todas as criações em questão, visando o atendimento dos requisitos e o aproveitamento das informações constantes em bancos de dados. Verifica-se a seguir os trabalhos correlatos, com embasamento em banco de dados orientados a objetos, modelagem lógica de dados e manipulação de dados, que contribuem com os estudos na área de Engenharia de *Softwares* e Desenvolvimento de Sistemas.

7.1 DESENVOLVIMENTO DE *SOFTWARE* DE APOIO AO PROJETO [...]

Este trabalho de Tiberti (2003, p. 145-147), descreve o desenvolvimento de um *software* com atuação em um escopo mais amplo que os das ferramentas atuais, baseado a partir de uma estrutura de *framework* de aplicação orientada a objeto. Desta forma, este *software* proposto seria capaz de suportar a introdução de novos métodos, algoritmos e ferramentas gráficas para análise e formação dos diversos tipos de arranjos físicos existentes, à medida que forem necessário. Esta descrição retirada do resumo e conclusão de Tiberti, salienta a permissão aos desenvolvedores em manipular as informações adaptando aos arranjos necessários para o *software.* Com isto, observa-se a necessidade de manipulação de informação nos desenvolvimentos de *softwares.*

7.2 A MODELAGEM DE NEGÓCIO COM UML [...]

O artigo de Ito; Martini; Iochida (2006, p. 9), relata a modelagem de negócio com UML permitindo uma maior compreensão dos sistemas de informação que agregue valor ao negócio. Os artefatos gerados nesta modelagem permitem uma validação precoce do sistema a ser desenvolvido e uma clara visão do negócio, além de mecanismos padronizados para definir, de forma adequada, os requisitos do sistema, aplicadas a técnicas de modelagem de negócio numa central de monitoração de diabéticos. Neste estudo, o artigo apresenta a viabilidade do desenvolvimento de *software* adaptado às necessidades dos clientes.

7.3 MÉTODO PARA ESPECIFICAÇÃO E MODELAGEM [...]

A dissertação de Panigassi (2007, p. 119-120), embora a complexidade do assunto, traz uma importante motivação pela necessidade de empresas de produção de *software* poderem fazer uso de um método que permitisse a definição de seus processos produtivos, onde o estudo estruturou uma proposta que viabilizasse a especificação e modelagem de uma fábrica de *software.* Como resultado o estudo apresentou um método baseado num modelo de processo conceitual induzido a partir do paradigma do processo de ciclo de vida de *software* com a utilização e adaptação ao domínio de processos do padrão *Open Distributing Processing (ODP),* o modelo *Capacity Maturity Model Integration (CMMI)* e o *Business Process Modeling Notation (BPMN).* Este estudo traz uma contribuição a um modelo de desenvolvimento de *software*, para o desenvolvimento de processos que compõe uma fábrica de *software*.

7.4 ENGENHARIA DE *SOFTWARE* PARA *SOFTWARE* LIVRE

Esta dissertação de Christoph (2004, p. 102-104), voltada à criação e desenvolvimento de *softwares* livres, permite um entendimento quanto as questões de evolução de *software.* Mostrou através de um protótipo, que seguindo o padrão proposto, pode produzir uma documentação que torna mais fácil o seu entendimento para novos participantes do projeto. A proposta apresentada neste trabalho visa diminuir esta barreira de entrada, e para tanto trabalha diretamente com a documentação do *software*. Essa proposta consiste de uma nova abordagem para documentar em termos da aplicação o código do sistema com base em cenários, de modo que estes fiquem encapsulados no código fonte. Compreende-se neste estudo o desenvolvimento de *software* com quebra de barreiras, na criação de documentações do *software*. Correlatos ao estudo em questão, faz jus às documentações do desenvolvimento de *softwares.*

7.5 FORMALISMOS ADAPTATVOS APLICADOS NA MODELAGEM [...]

A tese de Dizeró (2010, p. 115-117), buscou contribuir para o desenvolvimento de aplicações para *softwares* educacionais empregando formalismos adaptativos. Neste contexto, buscou-se a criação de um modelo genérico, capaz de permitir a criação de cursos para diferentes níveis de ensino através de diferentes práticas pedagógicas. Para isso foi elaborado um diagrama de classes, facilitando a codificação do *software*, onde o protótipo criado foi capaz de executar o modelo proposto de Máquina Moore Adaptativa, diante dos requisitos e modelagem de dados. A correlação do estudo se dá na possibilidade de se auto modelarem de acordo com as regras definidas pelo professor.

8 REPOSITÓRIO DE ARMAZENAMENTO DE MODELAGEM LÓGICA DE SISTEMAS

Tendo como finalidade a redução do tempo e custos na implantação de *softwares,* esta pesquisa tem como objetivo o estudo de três tipos de sistemas atuantes no mercado de *software*, dando enfoque principal ao sistema para indústria. Os demais sistemas, como o de saúde e comércio, embora façam parte da pesquisa e desenvolvimento, estarão dispostos no anexo e apêndices para análise comprobatória. (Ver Apêndices)

Em análise à relevância do tema, a pesquisa apresenta a identificação e modelagem lógica de aplicações já existentes, aprovadas e consolidadas no mercado, utilizando métodos e padrões homologados para o armazenamento e manipulação em um banco de dados objeto-relacional.

O protótipo permite que o desenvolvedor do *software,* execute as operações a partir de um banco de dados existente, uma vez que parte das informações e dos dados já estejam estruturados para utilização nos três tipos de sistemas, reduzindo assim o tempo de implantação e custos do projeto.

Alcançados os objetivos, o sistema utilizará e fará as modelagens a partir das informações, armazenadas em um Banco de Dados, demonstrando assim para o usuário ou adquirente do sistema uma visão macro do que está sendo disposto para execução em sua organização.

8.1 METODOLOGIA

Primeiramente, para o alcance dos objetivos e desenvolvimento do protótipo foi feito um estudo e levantamento das ferramentas a serem usadas. Desta forma, utilizou se da

ferramenta *Astah Community*, para a modelagem de dados, o *Microsoft SQL Server Express* como banco de dados e a linguagem de programação *Visual FoxPro*. Logo, conhecidas as ferramentas foi iniciada a implementação do protótipo, seguido dos testes necessários. Posteriormente foi implementado a funcionalidade de manipulação de um modelo existente (criado pela ferramenta Astah), assim dando a possibilidade de gravar num banco de dados.

8.2 RECURSOS UTILIZADOS

O *Astah Community* ferramenta utilizada para modelagens de dados foi utilizado como ferramenta externa para modelagem dos diagramas. Mesmo com limitações de uma versão grátis, a Astah Community trata-se de ferramenta adequada para modelagem UML.

8.3 PROTÓTIPO DESENVOLVIDO

O protótipo desenvolvido serve para manipulação de um repositório de modelagem lógica de sistemas desenvolvidos, testados e depurados, para serem usados na manutenção de sistemas em uso ou para a construção de novos. Para o desenvolvimento do protótipo, foram utilizadas as ferramentas *Microsoft SQL Server Express* e *Visual FoxPro*, as quais servem de suporte para armazenagem em banco de dados e linguagem de programação.

Microsoft SQL Server Express

Ferramenta utilizada para armazenagem de dados (Banco de Dados), de fácil utilização com um processo de instalação simples e robusto e com uma ferramenta gráfica que permitirá realizar a administração do servidor de uma forma simples e prática.

Visual FoxPro

Ferramenta utilizada como linguagem de programação. Escolhida pela praticidade e resposta eficaz para o trabalho desejado, como também melhor adaptada para o seu banco nativo Ms-SQL Server.

8.3.1 Repositório de Modelagem lógica

O sistema abaixo tem o intuído de manipular, ou seja, importar uma modelagem existente, para um repositório (Banco de dados). O sistema é constituído por duas telas, onde na tela principal conforme mostrado a baixo tem-se a opção de cadastro de um novo módulo, conforme o seu segmento. Na ilustração abaixo, visualiza-se 3 módulos cadastrados para o segmento Industrial (Estoque, Suprimento, Comercial).

Figura 17. Definição do Seguimento.

A Figura 17 dispõe dos botões para definição do seguimento desejado para a realização do trabalho. Neste caso, a opção será pela indústria.

Figura 18. Opção de Inserção ou Exclusão de Módulos.

Na criação da modelagem lógica, o usuário poderá acrescentar um novo módulo ou excluir o mesmo, conforme visualizado na Figura 18.

A Figura 19 ilustra as opções de salvar e sair, onde após a realização do trabalho ou toda alteração no sistema o usuário deverá salvar sua alteração.

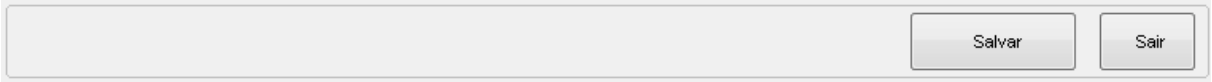


Figura 19. Opção Salvar.

Abaixo na Figura 20 o usuário poderá entrar no módulo selecionado, para edição dos diagramas.

Figura 20. Opção de Escolha de Módulos.

A Figura 21 demonstra a totalidade da tela de modelagem lógica.

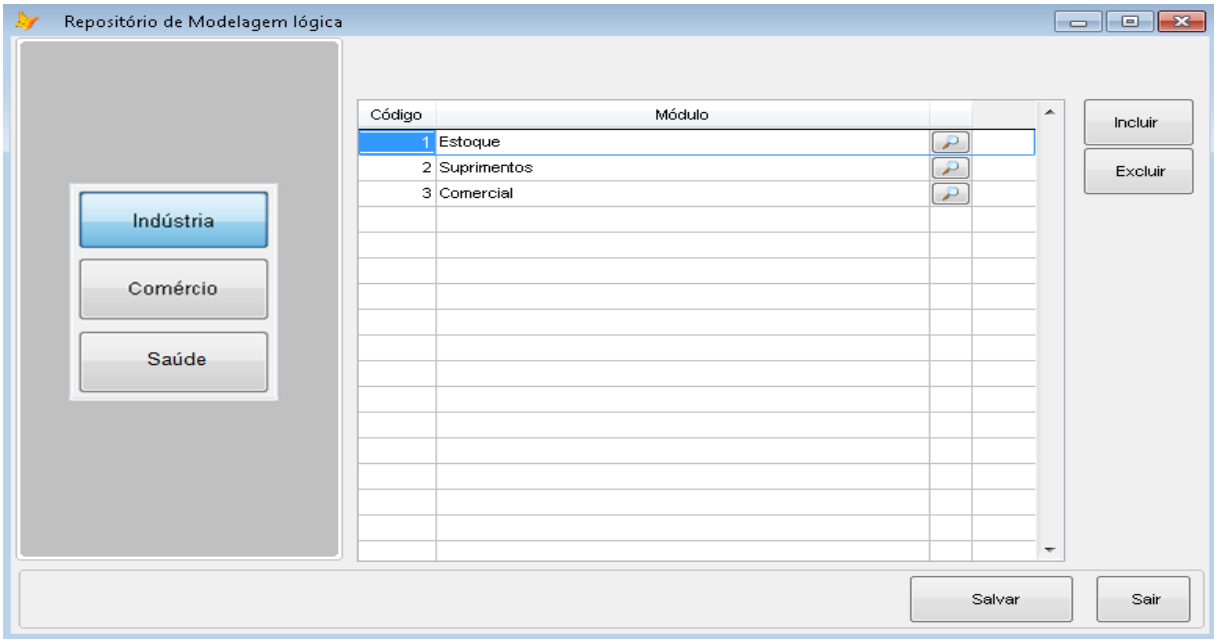


Figura 21. Repositório de Modelagem Lógica.

A tela a seguir informa ao usuário as três opções de criação de diagrama para inclusão no sistema.

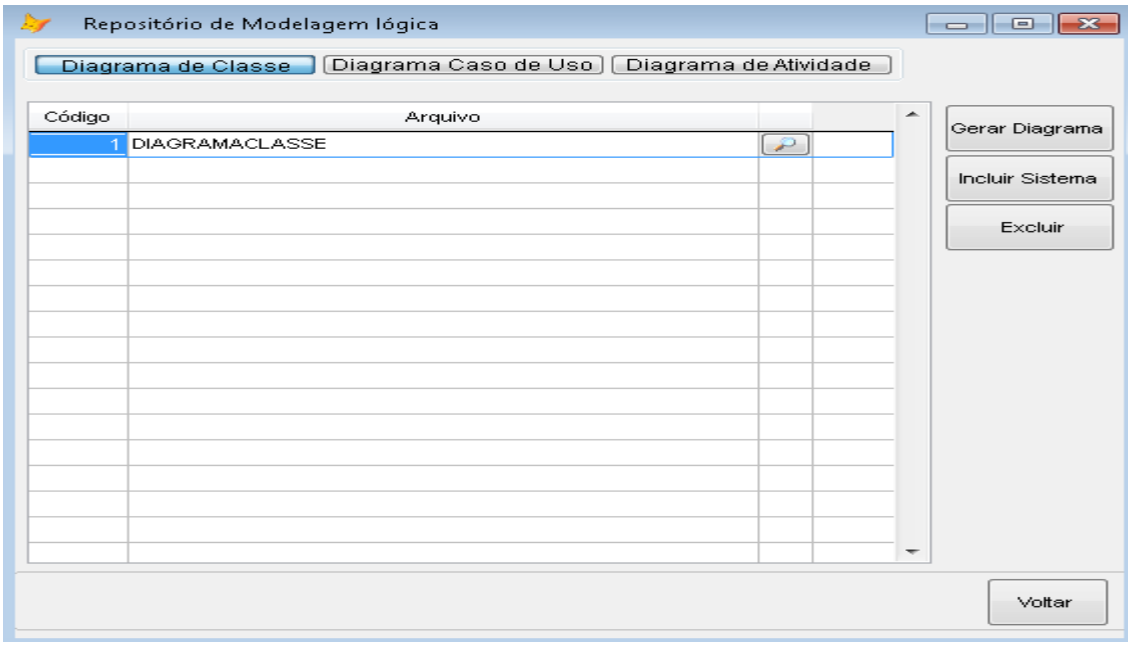


Figura 22. Opção da Seleção de Diagramas.

A Figura 22 possibilita ao usuário a escolha do diagrama desejado. Escolhido a opção, logo deverá gerar um diagrama ou utilizar um existente. Lembrando que o sistema apenas converte o arquivo e armazena em um repositório, assim possibilita que o usuário manipule esta informação.

Gerado o diagrama o usuário, deverá incluir no sistema, clicando no botão **Incluir Sistema** aparecerá uma nova tela para o usuário procurar onde seu arquivo foi salvo, assim o sistema fica com suas informações no grid. Logo após a inclusão o usuário apenas deve salvar suas alterações. Lembrando que essa consulta fica disponível em um banco de dados.

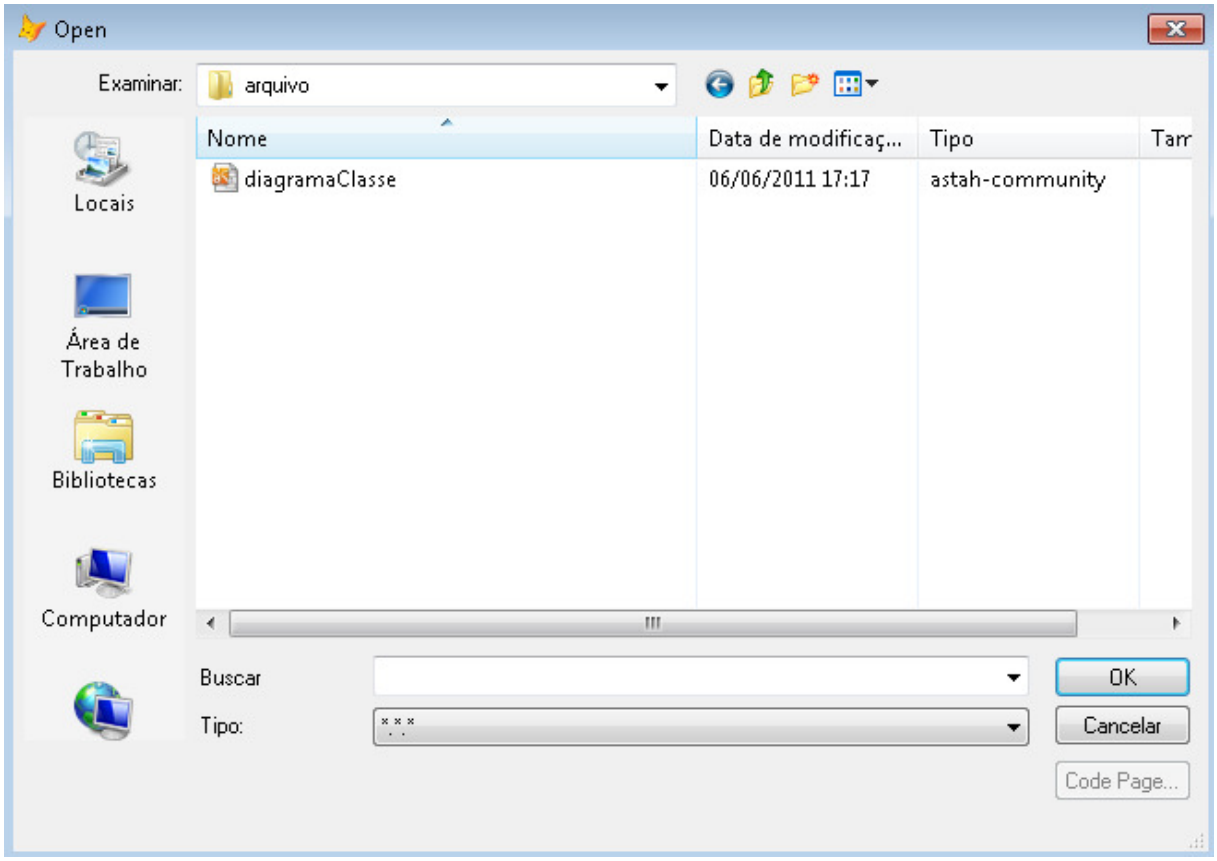


Figura 23. Localização de Arquivos.

Para implementação do projeto foi utilizada Modelagem UML com a ferramenta (ASTAH), o repositório utilizado foi Microsoft SQL Server 2005 Express. Linguagem de programação Microsoft Visual FoxPro 9.0 SP1.

O arquivo gerado e salvo, só é permitido ser editado por um programa externo, chamado Astah Community 6.3, exemplificado na Figura 24.

Figura 24. Astah Community 6.3.

8.4 RESULTADOS OBTIDOS

Os resultados obtidos através da criação deste repositório referem-se à facilidade dos engenheiros de *software* e clientes terem acesso ao acervo, onde poderão fazer consultas de modelos já existentes. Desta forma, esta consulta auxiliará na implantação do novo sistema, reduzindo o tempo real de uma implantação e também diminuindo os custos do projeto.

Para demonstração do protótipo foram catalogadas no repositório a modelagem dos sistemas nas áreas de: Saúde, Comércio e Indústria que podem ser manipulados e usados na manutenção de sistemas em funcionamento ou na criação de novos sistemas. A seguir serão mostrados os recursos de modelagens lógicas que poderão ser manipulados.

8.4.1 Funcionalidades

Para visualizar as funcionalidades serão usados Diagramas de Casos de Uso. O diagrama de caso de uso a seguir apresenta as funcionalidades do sistema de Industrial.

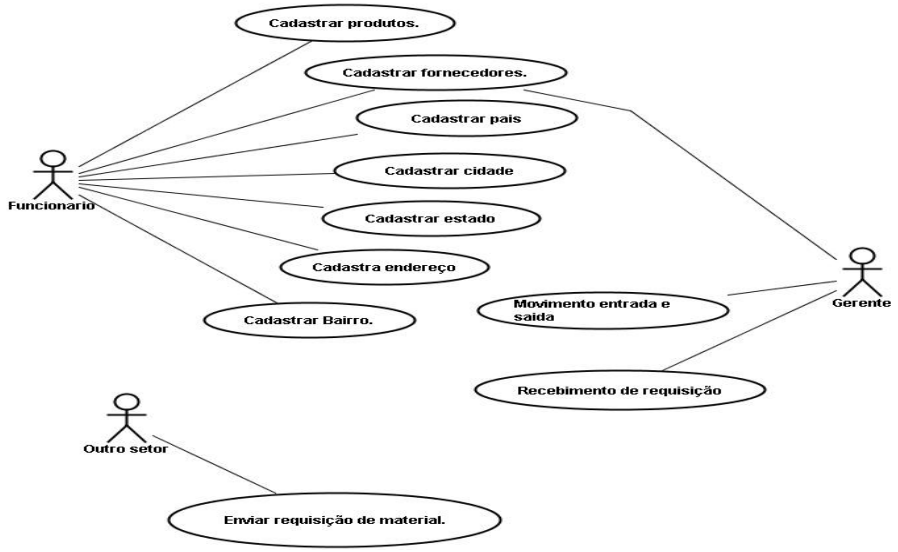


Figura 25. Diagrama de Casos de Uso

Os demais sistemas saúde e comercial estão demonstrados nos anexos.

8.4.2 Armazenamento

A modelagem do banco de dados pode ser representada em UML pelo diagrama de classe entidade, além de servir também para especificação do dicionário de dados. O diagrama de classes entidades a seguir apresenta visualmente as relações e os modelos de objetos do sistema de Industrial.

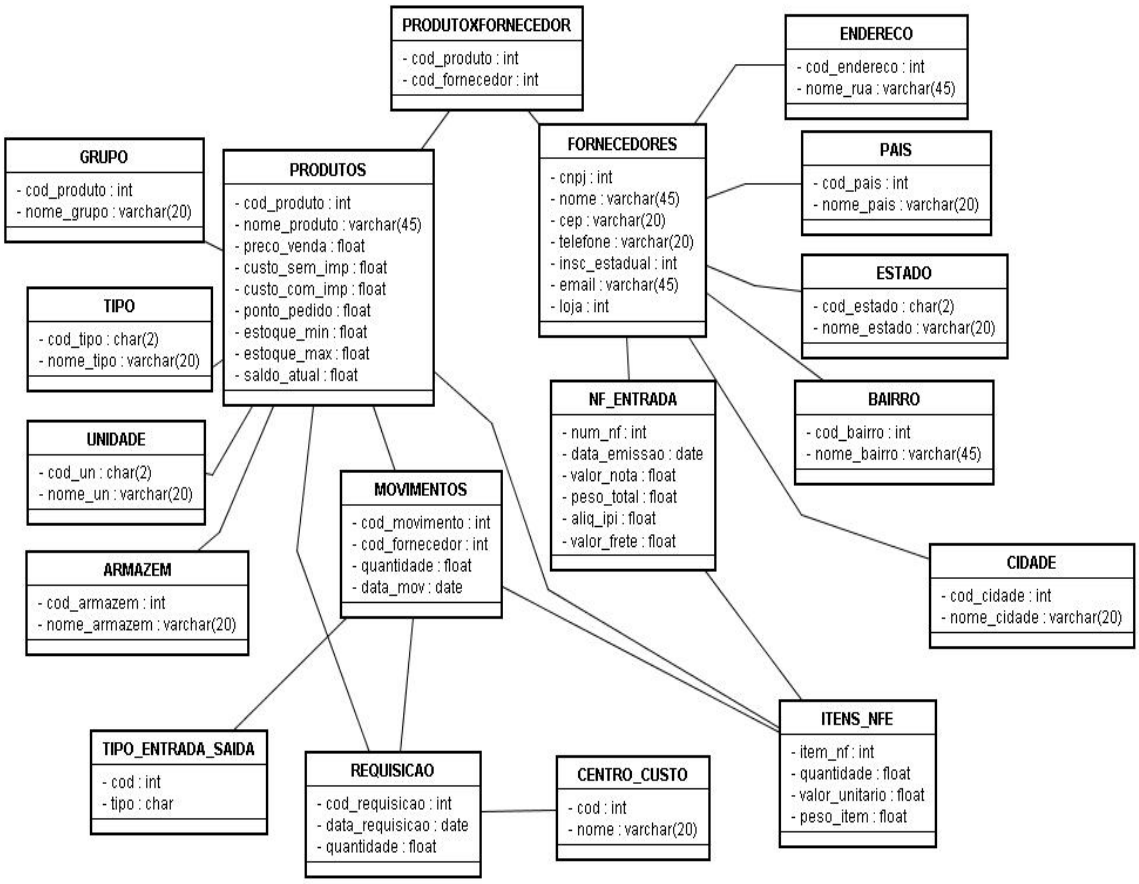


Figura 26. Diagrama de Classes

Os demais sistemas saúde e comercial estão demonstrados nos anexos.

8.4.3 Fluxo de Dados

Para apresentar o fluxo e as ações, (caminho, cálculos, decisões etc) das atividades do sistema de Industrial será apresentado o diagrama da atividade de cadastrar fornecedores.

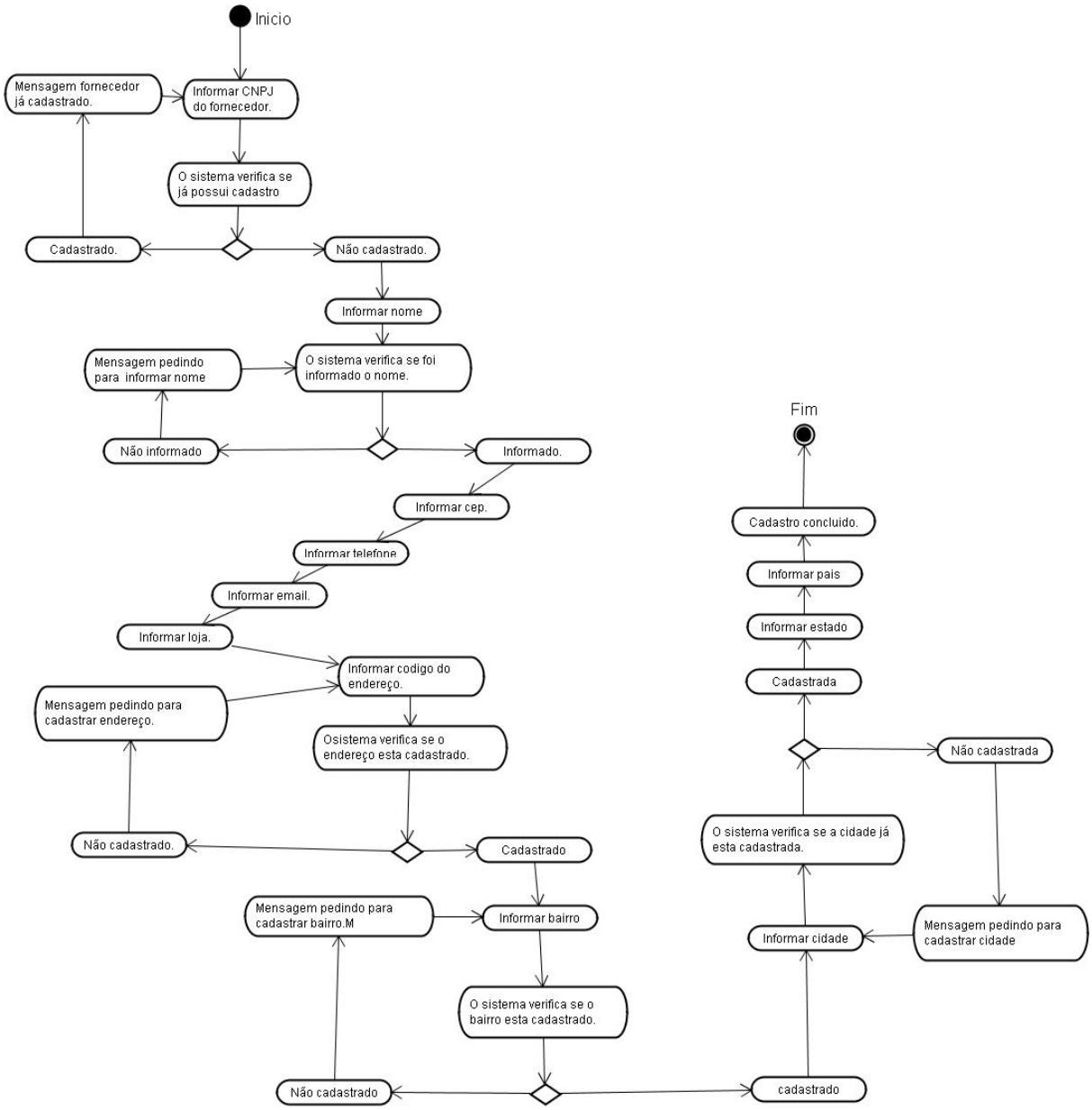


Figura 27. Diagrama de Atividades – Cadastro de Fornecedores

Os demais sistemas saúde e comercial estão demonstrados nos anexos.

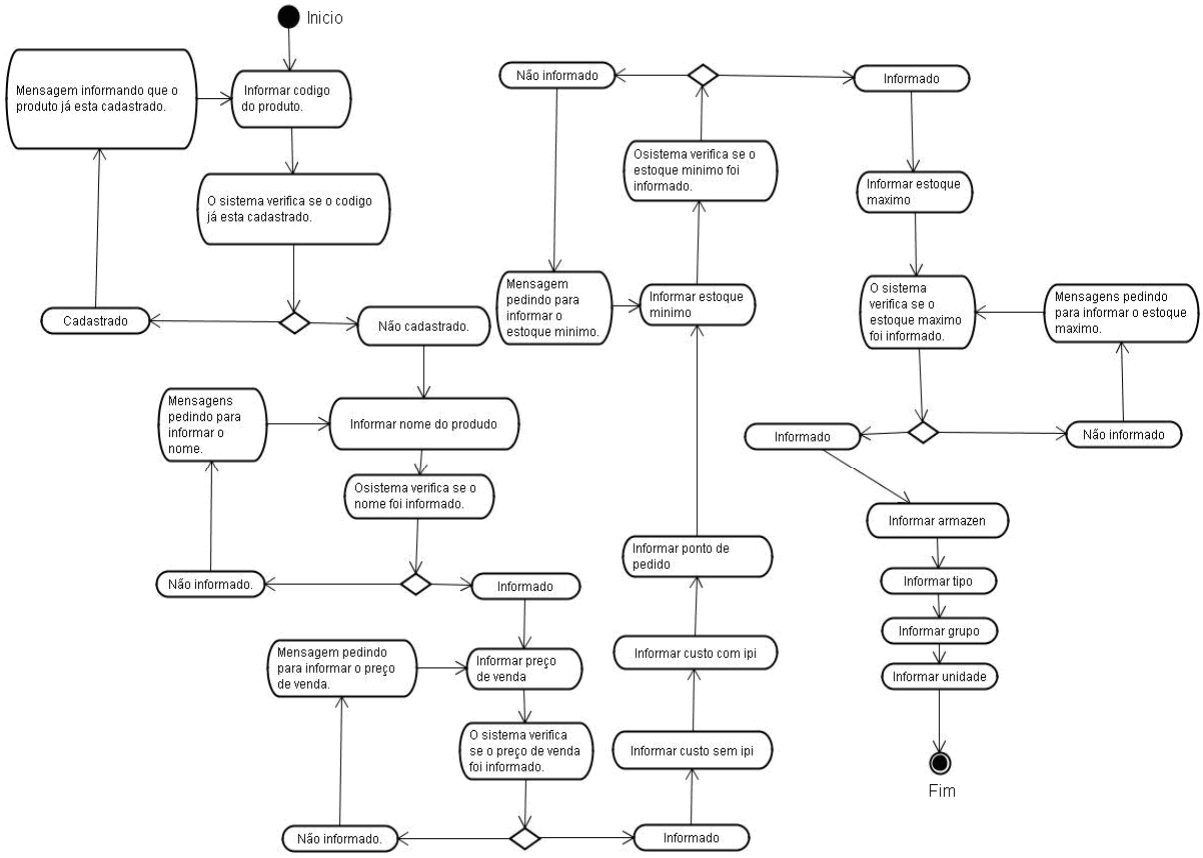


Figura 28. Diagrama de Atividades – Cadastro de Produtos

8.4.4 Dicionário de Dados

O dicionário de dados, trata-se das informações sobre as tabelas de programação, onde para cada atividade do sistema, ou campos a serem preenchidos e processadas as informações tem sua respectiva tabela. Os quadros apresentados a seguir, mostram as tabelas referentes ao cadastro de produtos, e informações dos fornecedores, do sistema de Industrial.

PRODUTO – Tabela que contém as informações dos produtos.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**COD_PRODUTO**	**INTEGER**	NN	Código produto, Único, maior que 0, auto incremento
FK	UNIDADE_COD_UN	CHAR(2)	NN	Código de unidade medida (pc, m2, un, cx)
FK	TIPO_COD_TIPO	CHAR(2)	NN	Código do tipo produto (mp(Matéria prima), pa(Produto Acabado), pi(Produto Intermediário)).
FK	ARMAZEM_COD	INTEGER	NN	Código do armazém produto, maior que 0
FK	GRUPO_COD_GRUPO	INTEGER	NN	Código do grupo que o produto pertence, maior que 0
	NOME_PRODUTO	VARCHAR(45)	NN	Nome do Produto
	PRECO_VENDA	FLOAT	NN	Valor preço de venda, com 2 casas decimais, maior que 0
	CUSTO_SEM_IMP	FLOAT		Valor preço sem impostos, com 2 casas decimais, maior que 0
	CUSTO_COM_IMP	FLOAT		Valor preço com impostos, com 2 casas decimais, maior que 0
	PONTO_PEDIDO	FLOAT		Quantidade à repor no estoque, com 2 casas decimais, maior que 0
	ESTOQUE_MIN	FLOAT	NN	Quantidade mínima em estoque, com 2 casas decimais, maior que 0
	ESTOQUE_MAX	FLOAT	NN	Quantidade mínima em estoque, com 2 casas decimais, maior que 0
	SALDO_ATUAL	FLOAT		Quantidade atual em estoque, com 2 casas decimais, maior que 0

Quadro 21. Tabela de Produtos.

FORNECEDOR – Tabela que contém dados pessoais dos fornecedores dos produtos.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**CNPJ**	**VARCHAR(18)**	NN	CNPJ do Fornecedor, Único, formato "99.999.999/9999-99"
FK	ENDERECO_COD_ENDERECO	INTEGER	NN	Código do endereço , 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
	NOME	VARCHAR(45)	NN	Nome do Fornecedor
	TELEFONE	VARCHAR(13)		Telefone do Cliente, formato "(99)9999-9999"
	EMAIL	VARCHAR(45)		E-mail do Fornecedor
	LOJA	INTEGER		Número filial
	CEP	VARCHAR(9)		CEP formato "99999-999"
	INSC_ESTADUAL	VARCHAR(11)		Inscrição estadual do fornecedor, Único, formato "999.999.999"

Quadro 22. Tabela de Fornecedores.

As demais tabelas estão demonstradas nos anexos.

CONCLUSÃO

Com este trabalho, foi possível contribuir para a evolução das técnicas de modelagem, em especial o desenvolvimento de um protótipo para manipulação destas modelagens, de uma forma estruturada em uma base de dados (repositório). Bem como o seu entendimento, a partir das ferramentas utilizadas para o desenvolvimento.

O embasamento baseado no levantamento e estudo bibliográfico proporcionou um acúmulo de conhecimento para o tratamento de modelagem lógica de sistemas, com enfoque na engenharia de *software*, engenharia de requisitos e banco de dados. Os quais consistem nas formas de desenvolvimento e armazenagem de dados. Ainda neste sentido, a literatura proporcionou o conhecimento dos modelos de desenvolvimento de *softwares*, como também, arquitetura e gerenciamento de banco de dados.

Outro propósito atingido foi a identificação das modelagens lógicas no desenvolvimento dos *softwares* estudados, possibilitando catalogar suas informações utilizando a linguagem de modelagem UML. E por fim, fornecer estas informações por meio de um protótipo, para futuras consultas.

Houve dificuldade para criar o protótipo no que se refere a apresentação dos modelos lógicos, ou seja, os diversos diagramas para visualização do sistema. Para resolver este problema foi utilizada a ferramenta Astah a qual manipula os modelos armazenados no repositório pelo protótipo.

A construção do protótipo contou com a utilização da ferramenta *Astah Community*, para a modelagem de dados, o *Microsoft SQL Server Express* como banco de dados e a linguagem de programação Visual FoxPro.

De forma geral, pode-se concluir que o desenvolvimento de *software* ganha um novo aliado no sentido de aumento de qualidade, redução de tempo no desenvolvimento,

consequentemente diminuição de custos. Isto porque com um repositório de modelos lógicos de aplicações, as fases de levantamento, analise, modelagem e especificação de requisitos,dos modelos prontos obtidos no repositório, são eliminadas. Assim, sugere-se como trabalhos futuros:

a) Incluir no protótipo funcionalidades para manipulação de modelos lógicos para substituir a ferramenta externa (Astah);
b) Agregar ferramentas CASE para implementação do protótipo;
c) Desenvolvimento de um software, baseado na ontologia.

REFERÊNCIAS

ALEXANDRUK, Marcos. **Modelagem de banco de dados.** 2011. Disponível em: www.unilivros.com.br/pdf/dbmod.pdf>. Acesso em: 22 mai. 2011.

ARAÚJO, Dinaldo do Nascimento. Modelo lógico de banco de dados proposto para uma contabilidade de partida múltipla. **Adcontar**, Belém, v.4, n.2, p.27-44, nov. 2003.

BANKI, André Luis; TANAKA, Sérgio Akio. Metodologias ágeis: uma visão prática. **Engenharia de *Software* Magazine**, Rio de Janeiro, ano 1, ed. 4, p. 22-29, 2008.

BERWANGER, Elielder. **Apostila Engenharia de *Software*.** 2009. Disponível em : < http:// www.fag.edu.br/professores/**elielder/**materias/.../**apostila**_eng_**software**.pdf >. Acesso em: 20 mai. 2011.

BEZERRA, Eduardo. **Princípios de analise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2003. 286 p. Disponível em: < http://wiki.sj.ifsc.edu.br/wiki/index.php /Ciclo_de_Vida_Iterativo_e_Incremental>. Acesso em: 20 mai. 2011.

BRANCO, Renata. Automação comercial. 2010. Disponível em: http://www.manutencaoe suprimentos.com.br/conteudo/1802-automacao-comercial/. Acesso em: 12 jun. 2011.

CAMARGO, Liriane Soares de Araújo de. **Metodologia de desenvolvimento de ambientes informacionais digitais a partir dos princípios da arquitetura da informação**. 2009. 287 f. Tese (Doutorado em Ciências da Informação) – Faculdade de Filosofia e Ciências – Universidade Estadual Paulista, Marília.

CHRISTOPH, Roberto de Holanda. **Engenharia de *software* para *software* livre.** 2004. 118 f. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

DIAS, Adélio André. **Análise de sistemas I**. 2002. Disponível em: < www.di.ubi.pt /~ddg/aulas/licenciatura/asi/asi/chapter8.pdf >. Acesso em: 25 mai. 2011.

DIZERÓ, Wagner José. **Formalismos adaptativos aplicados na modelagem de *softwares* educacionais.** 2010. 127 f. Tese (Doutorado em Engenharia Elétrica e Sistemas Digitais) – Escola Politécnica da Universidade de São Paulo, São Paulo.

FALBO, Ricardo de Almeida. **Engenharia de Requisitos**. Espírito Santo, 2011. UFES - Universidade Federal do Espírito Santo. Disponível em: < www.inf.ufes.br /~falbo/files/Notas_Aula_Engenharia_Requisitos.pdf >. Acesso em: 14 mai. 2011.

FERRARI, Thales Martins. **Sistema de *Software* para auxílio dos docentes da Unifei**. 2006. 138 f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Itajubá, Minas Gerais.

FOWLER, Martin. **A nova metodologia**. 2003. Disponível em: < http://simplus .com.br /artigos/ a-nova-metodologia/#version-list>. Acesso em: 20 mai. 2011.

GASTALDO, Denise Lazzeri; MIDORIKAWA, Edson Toshimi. Processo de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho: um estudo de caso. In: INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING., Piracicaba, SP, 2004. **WER03 proceedings.**. Rio de Janeiro: PUC, p. 302-328. 2004.

GOMIDE, Fernando. **Fundamentos de modelagem de sistemas**. 2007. DCA-FEEC-Unicamp. Curso IA 881 Otimização Linear da Faculdade de Engenharia Elétrica e de Computação. Disponível em: < www.dca.fee.unicamp.br/~gomide /.../IA881/.../IA881Modelagem.pdf>. Acesso em : 12 mai. 2011.

HERINGER, Aílson F.. O Visual FoxPro como Ferramenta de Manipulação de Dados. 2004. Disponível em: < http://www.vfpbrasil.com.br/modules/newbb/viewtopic.php? topic_id=373&forum=16. Acesso em: 05 jun. 2011.

HEUSER, Carlos Alberto. **Projeto de banco de dados.** 4. ed Porto Alegre: Sagra Luzzatto, 2001. 204 p.

ITO, Márcia; MARTINI, José Sidnei C.; IOCHIDA, Lúcia C. A modelagem de negócio com UML de uma central de monitoração de diabéticos. **RESI-Revista Eletrônica de Sistemas de Informação,** São Paulo, ed. 9, n. 3, 2006.

JUNQUEIRA, Fabrício. **Modelagem e simulação distribuídas de sistemas produtivos**. 2006. 222 f. Tese (Doutorado em Engenharia Mecatrônica e de Sistemas Mecânicos). Escola Politécnica da Universidade de São Paulo, São Paulo.

LANGA, Sara Alvarez. **Modelo de bancos de dados**. 2007. Disponível em: < http:// www.criarweb.com/artigos/modelos-banco-dados.html >. Acesso em: 25 mai. 2011.

LESSA, Rafael Orivaldo; LESSA JUNIOR, Edson Orivaldo. **Modelos de processos de engenharia de *software***. Palhoça, 2010. Disponível em: < http://ebookbrowse. com/search.php?q=rafael+orivaldo+lessa>. Acesso em: 13 mai. 2011.

LUDVIG, Diogo. REINERT, Jonatas Davson. **Estudo do uso de Metodologias Ágeis no Desenvolvimento de uma Aplicação de Governo Eletrônico**. 2007. Departamento de Informática e Estatística – Universidade Federal de Santa Catarina Disponível em: < http://projetos.inf.ufsc.br/arquivos_projetos/projeto_589 /Artigo_Diogo_Jonatas.pdf >. Acesso em: 18 mai. 2011.

MACORATTI, José Carlos. **Conceitos básicos de modelagens de dados**. Disponível em: < http://www.macoratti.net/cbmd1.htm >. Acesso em: 21 mai. 2011.

MAZZOLA, Vitório Bruno. **Engenharia de *Software***: Conceitos Básicos. Florianópolis, 2010. INE/CTC/UFSC. Disponível em: < http://algol.dcc.ufla.br/~monserrat/icc/ Introducao_ES.pdf >. Acesso em 15 mai. 2011.

OLIVEIRA, Cleiane Gonçalves; BARROS, Kátia Adriana Alves Leite de; OLIVEIRA, Ariane Gonçalves de. **J. Health Inform.** Minas Gerais. p. 1-6, jan-mar. 2010.

OLIVEIRA, Luiz Ângelo. **Ciclo de vida de sistemas de informação**. São Paulo, 2009. Disponível em: < http://www.eteavare.com.br/arquivos/43_37.pdf >. Acesso em: 14 mai. 2011.

PANIGASSI, Rogério. **Método para especificação e modelagem de processos de fábrica de *software* usando RM-ODP e BPM**. 2007. 152 p. Dissertação (Mestrado em Engenharia de Sistemas Digitais) – Escola Politécnica da Universidade de São Paulo, São Paulo.

PARREIRA JUNIOR, Walteno Martins. **Engenharia de *Software***. Universidade do Estado de Minas Gerais. 2011. 108 f. Apostila - Fundação Educacional de Ituiutaba. Curso de Engenharia da Computação e Engenharia de *Software*. Disponível em: < http://www.waltenomartins.com.br/ap_es_v1.pdf>. Acesso em: 13 mai. 2011.

PENDER, Tom. **UML, a bíblia.** Rio de Janeiro: Elsevier, 2004. 711 p.

PINHEIRO, Nilton. **Conhecendo o SQL Server 2005 Express Edition**. 2006. Disponível em: < http://www.linhadecodigo.com.br/artigo/947/Conhecendo-o-SQL-Server-2005-Express-Edition.aspx>. Acesso em: 05 jun. 2011.

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

QUEIROZ, Carlos Octávio de Alexandre. **Modelo de gestão do conhecimento para empresas de desenvolvimento de *software***. 2001. 153 f. Dissertação (Mestrado em Informática) – Universidade Federal da Paraíba, Campina Grande.

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de *software***. 2007. 139 f. Dissertação (Mestrado em Ciências da Computação) – Universidade de São Paulo, São Paulo.

SAVI, Antonio Francisco. **Modelo de sistema para gerenciamento de conhecimentos explícitos em abordagens de DFA (Design For Assembly).** Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

SILVA, Cristiano Campos; SOUZA, Kleyton Farias de; DANTAS, Samuel Dias. **Metodes: Metodologia de desenvolvimento de *software***. 2006. 159 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Faculdade Cenecista de Brasília. Distrito Federal.

SILVA, Luís César. **Simulação de Processos.** Universidade Federal do Espírito Santo. 2005. Disponível em: < http://www.agais.com/simula.htm>. Acesso em: 10 mai. 2011.

SILVEIRA, Daniel Gomes; ALVES, Filipe Ferreira; FERREIRA, Leandro Soriano; SOUZA, Rodrigo Rocha Gomes e. Banco de dados orientados a objetos-relacionais. 2006. Departamento de Ciências da Computação da Universidade Federal da Bahia. Disponível em: < http://im.ufba.br/pub/MATA60/SemestreLetivo20062/ Bancos_de_dados_OO_e_OR.pdf >. Acesso em: 28 mai. 2011.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de *Software***. DCC/UFLA – RCC/Infocomp. Vol. 3, n. 2. Minas Gerais,

nov. 2004. Artigos. Disponível em: < http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf >. Acesso em: 13 mai. 2011.

SOUZA, Cynara Lira Carvalho de. **Modelagens de Sistemas**. 2006. Faculdade de Ciências Sociais e Aplicadas de Petrolina – FACAPE. Teoria Geral de Sistemas - TGS.Disponível em: < www.facape.br/cynara/.../MODELAGEM_DE_SISTEMAS_ aula03.ppt >. Acesso em: 10 mai. 2011.

SOUZA, Luiz Gustavo S. de. **Preparando um ambiente de desenvolvimento Java EE baseado em Eclipse**. 2009. Disponível em: < http://luizgustavoss.wordpress.com/tag/astah-community/ >. Acesso em: 05 jun. 2011.

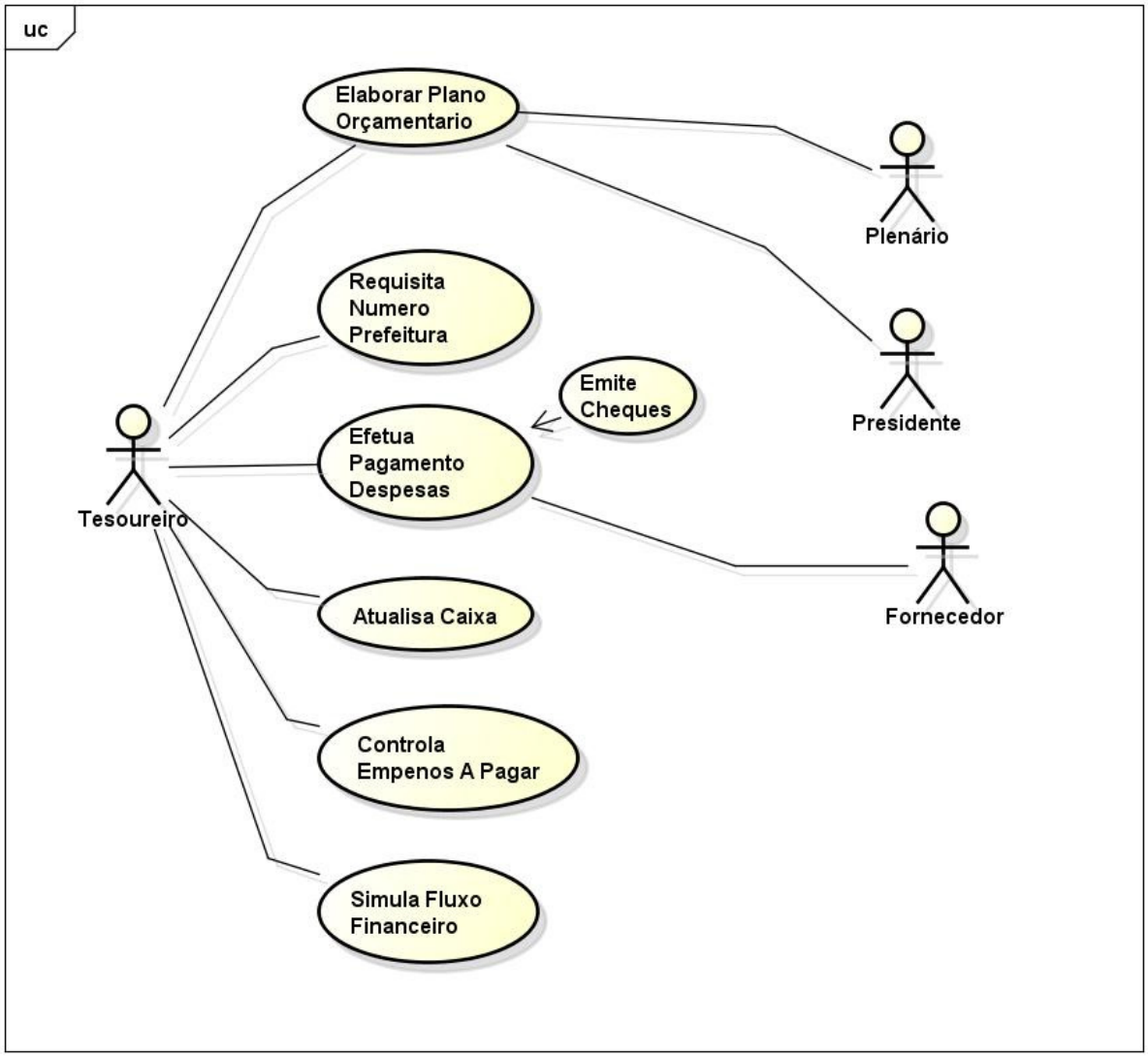
SOMMERVILLE, Ian. **Engenharia de *software*/** Ian Sommerville. São Paulo: Addison-Wesley, 2003. 592 p.

TAKAI, Osvaldo Kotaro; ITALIANO, Isabel Cristina; FERREIRA, João Eduardo. **Introdução a Banco de Dados.** 2005. DCC-IME-USP. Disponível em: < www.ime. usp.br/~jef/apostila.pdf>. Acesso em: 25 mai. 2011.

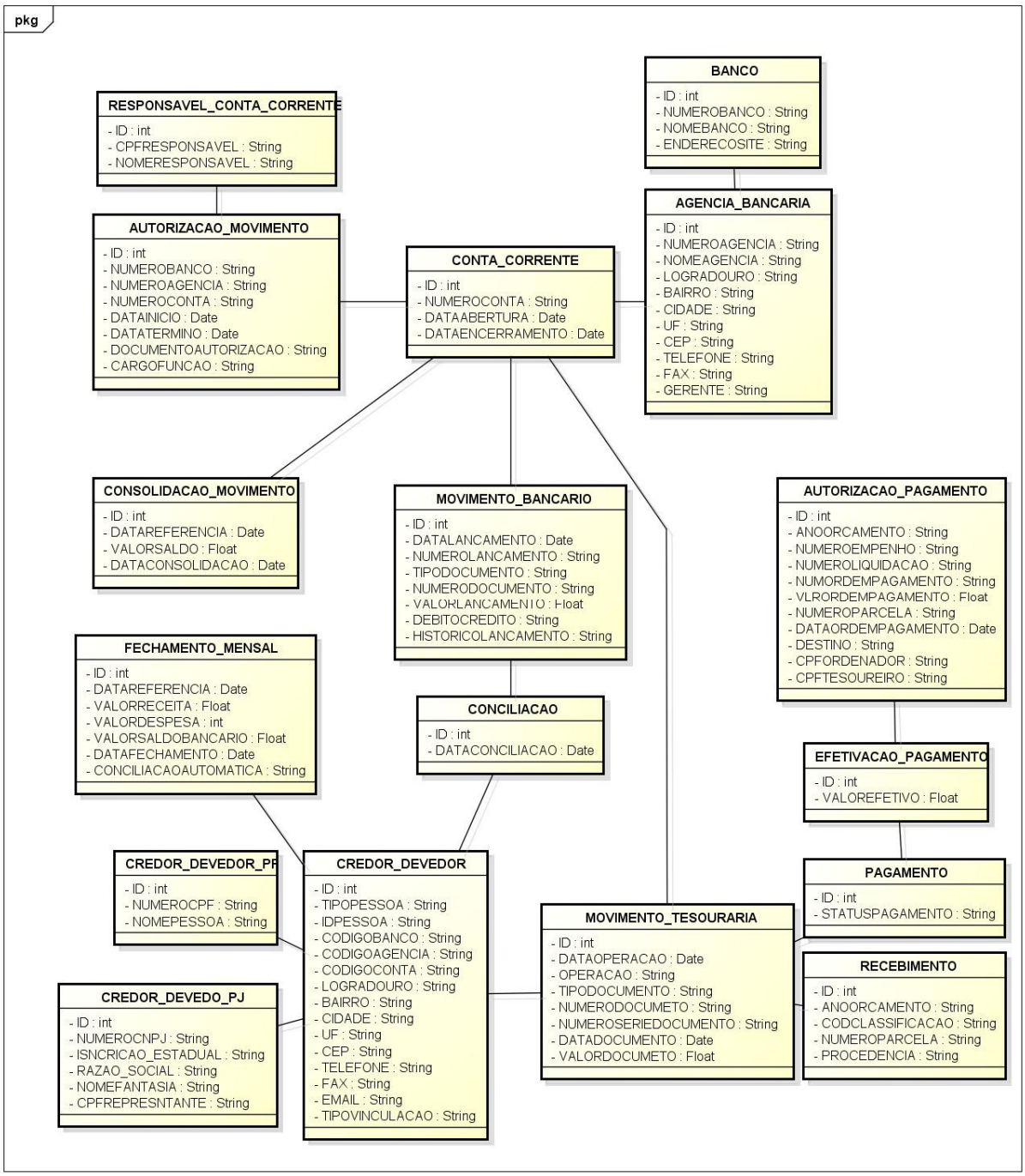
TIBERTI, Alexandre José. **Desenvolvimento de *software* de apoio ao projeto de arranjo físico de fábrica baseado em um *framework* orientado a objeto**. 182 f. 2003. Tese (Doutorado em Engenharia Mecânica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

APÊNDICES

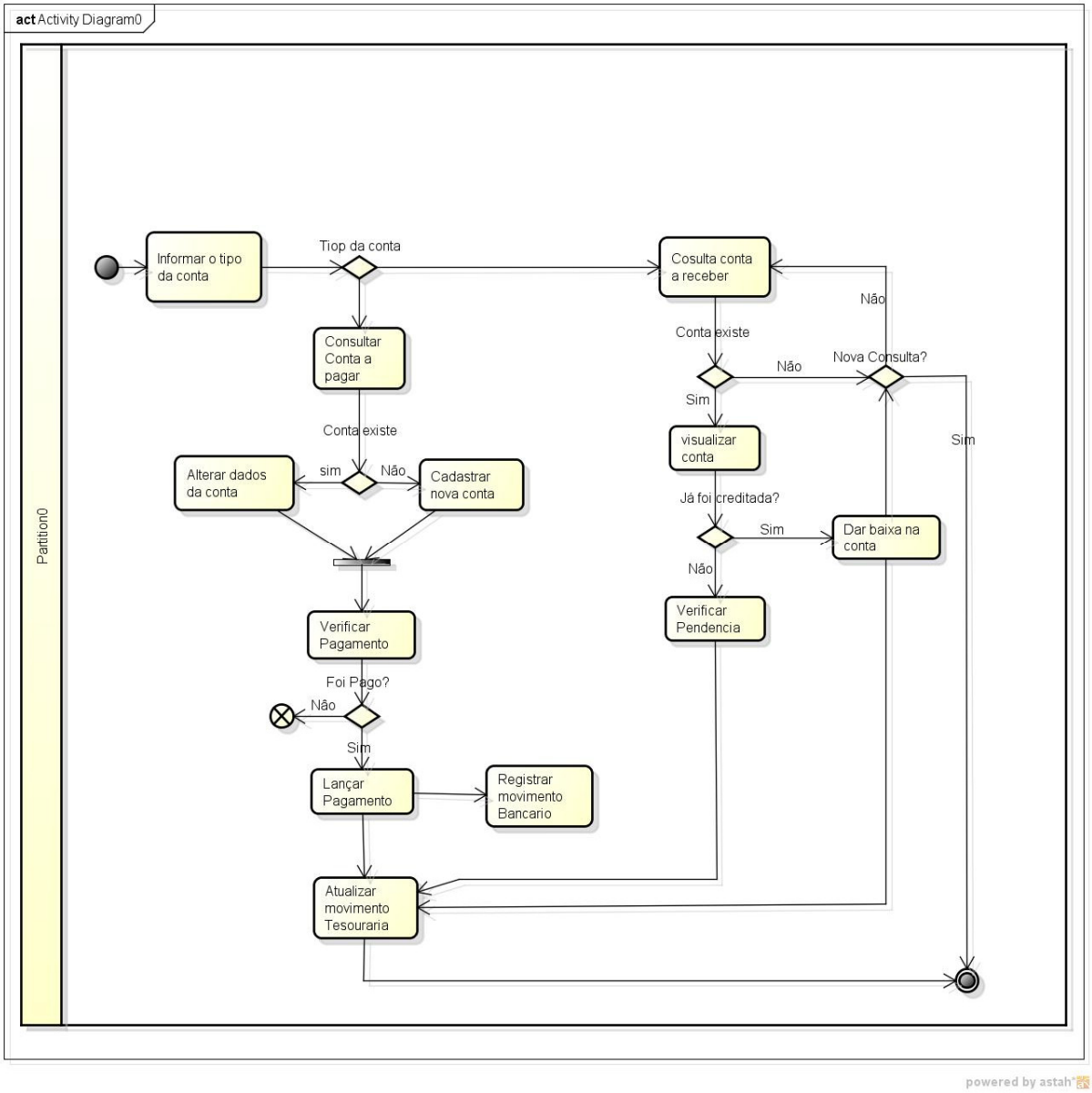
APÊNDICE A – Diagrama de Caso de Uso Tesouraria/Comércio



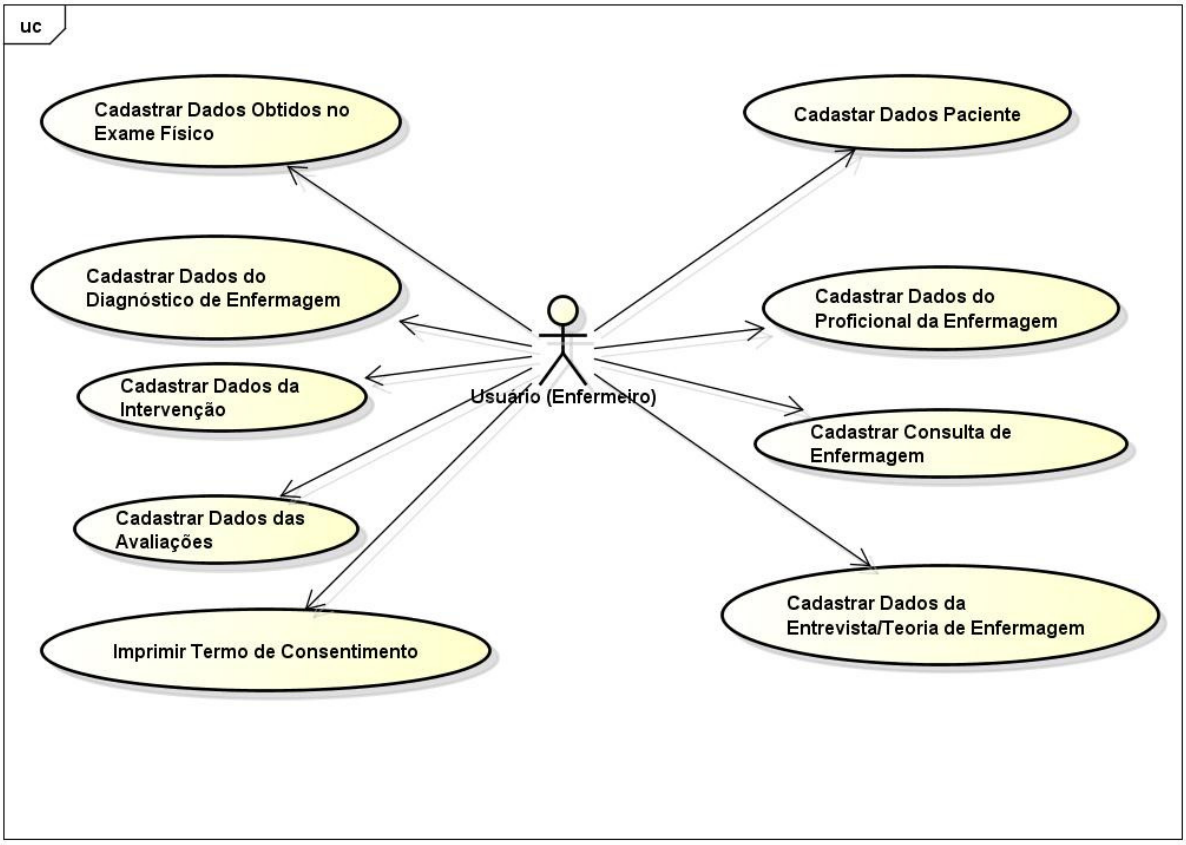
APÊNDICE B – Diagrama de Classe Tesouraria/Comércio



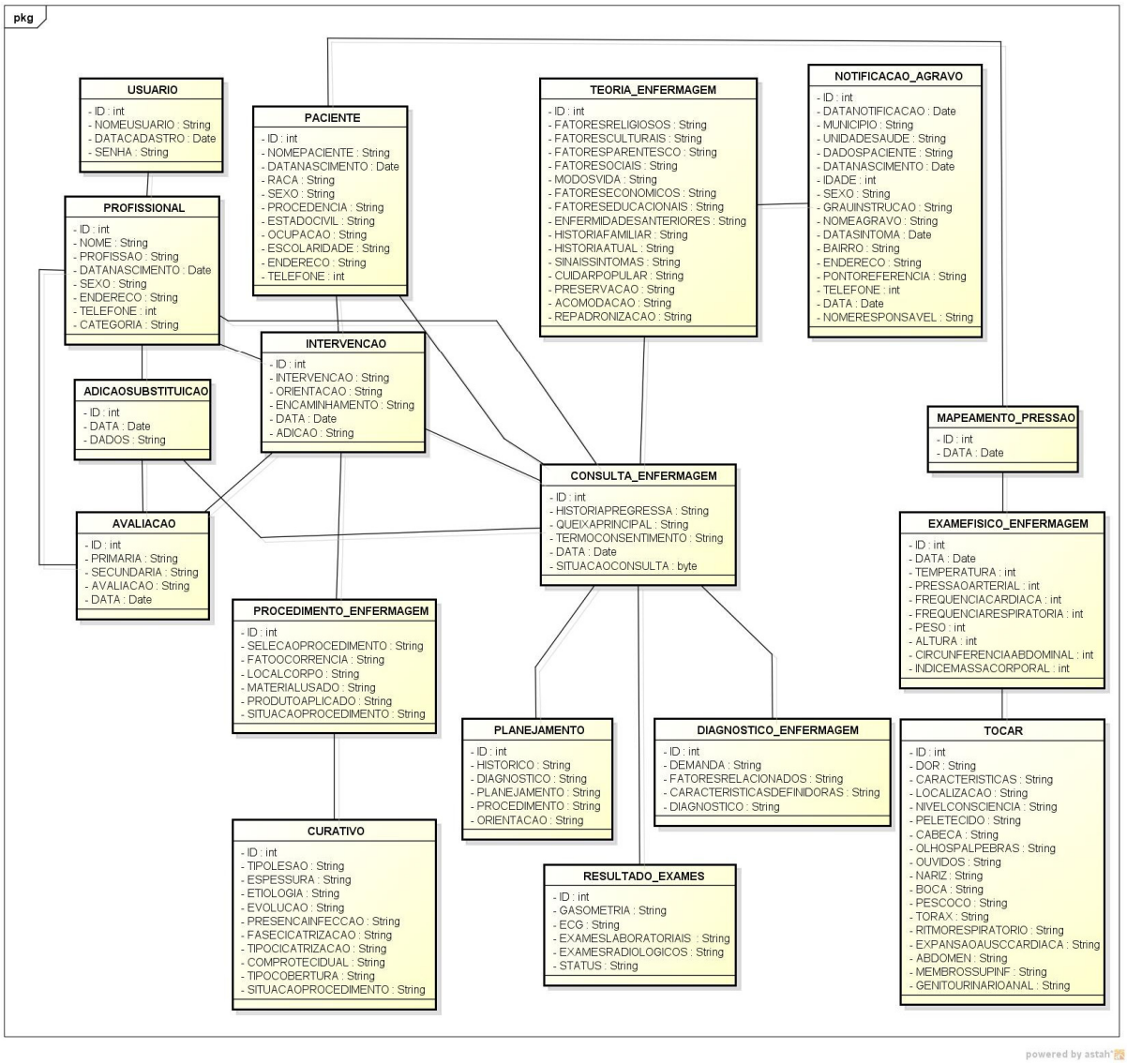
APÊNDICE C – Diagrama de Atividades Tesouraria/Comércio



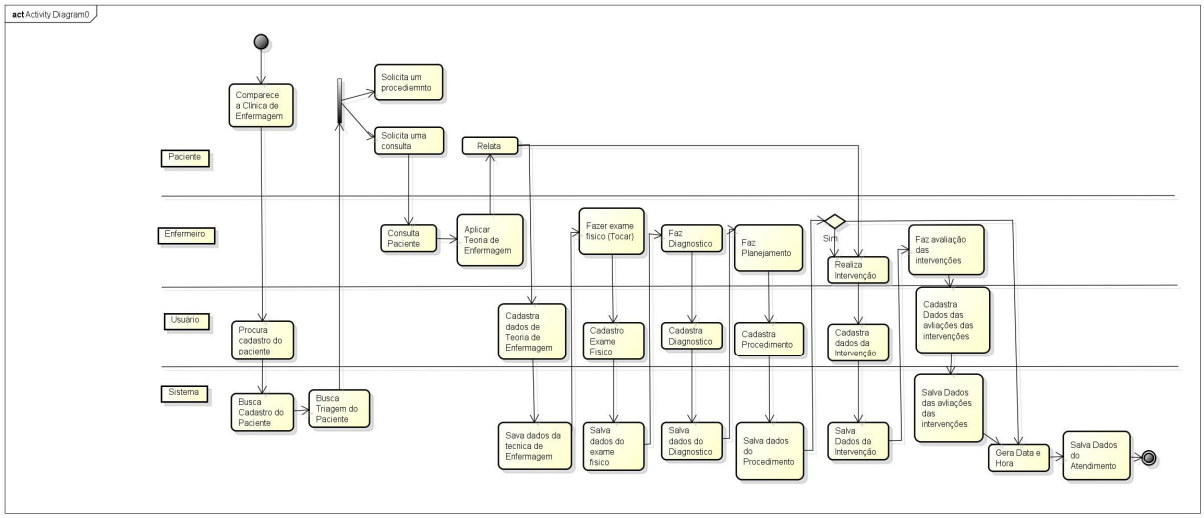
APÊNDICE D – Diagrama de Caso de Uso Enfermagem/Saúde



APÊNDICE E – Diagrama de Classe Enfermagem/Saúde



APÊNDICE F – Diagrama de Atividades Enfermagem/Saúde



APÊNDICE G – 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
	NOME_BAIRRO	VARCHAR(45)	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
	NOME_CIDADE	VARCHAR(45)	NN	Nome da Cidade

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
	NOME_ESTADO	VARCHAR(45)	NN	Nome do Estado

País - 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
	NOME_PAIS	VARCHAR(20)	NN	Nome do País

NF_ENTRADA - Tabela que contém dados de todas as notas fiscais de entrada.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**NUM_NF**	**INTEGER**	NN	Número do Pedido, Único, maior que 0
FK	FORNECEDORES_CNPJ	VARCHAR(18)	NN	CNPJ do Fornecedor, Único, formato “99.999.999/9999-99”
	VALOR_NOTA	FLOAT	NN	Valor total da nota fiscal , com 2 casas decimais, maior que 0
	DATA_EMISSAO	DATE	NN	Data Da nota fiscal de entrada, formato DD/MM/AAAA
	PESO_TOTAL	FLOAT		Peso Total do NFE, com 2 casas decimais
	ALIQ_IPI	FLOAT		Quantidade IPI do Produto, com 2 casas decimais
	VALOR_FRETE	FLOAT		Valor Frete, com 2 casas decimais

ITENS_NFE - Tabela armazena os as informações das Notas Fiscais de entrada.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	ITEM_NF	INTEGER	NN	Número do Item, maior que 0
FK	PRODUTO_COD_PRODUTO	VARCHAR(18)	NN	Código do Produto, maior que 0
FK	NF_ENTRADA_NUM_NF	INTEGER	NN	Numero NFE, maior que 0, Unico
	PESO_ITEM	FLOAT	NN	Peso Item, com 2 casas decimais
	VALOR_UNITARIO	FLOAT	NN	Valor unitario, com 2 casas decimais
	QUANTIDADE	FLOAT	NN	Quantidade do Item, com 2 casas decimais

PRODUTO_has_FORNECEDORES - Tabela que contém informações de relacionamento das tabelas produto X Fornecedores.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
FK	**PRODUTO_COD_PRODUTO**	**INTEGER**	NN	Código do Produto, Único, maior que 0
FK	FORNECEDORES_CNPJ	INTEGER	NN	CNPJ do Fornecedor, Único, formato “99.999.999/9999-99”

ENDERECO – Tabela que contém informações sobre endereços dos fornecedores.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**COD_ENDERECO**	**INTEGER**	NN	Código do endereço, Único, maior que 0
	NOME_RUA	VARCHAR(45)	NN	Nome da Rua

REQUISICAO - Tabela que contém dados sobre as requisições .				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**COD_REQUISICAO**	**INTEGER**	NN	Código da Requisição, único, maior que 0, auto incremento
FK	CENTRO_CUSTO_COD	INTEGER	NN	Código Centro de Custo, maior que 0
FK	PRODUTO_COD_PRODUTO	INTEGER	NN	Código do Produto, maior que 0
FK	DATA_REQUISICAO	DATE	NN	Data requisição, formato DD/MM/AAAA
	QUANTIDADE	FLOAT	NN	Quantidade, com 2 casas decimais

MOVIMENTOS - Tabela que contém as informações das movimentações.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	**COD_MOVIMENTO**	**INTEGER**	NN	Código das Movimentações, único, maior que 0, auto incremento
FK	ITENS_NFE_ITEM_NF	INTEGER	NN	Número do Item, maior que 0
FK	REQUISICAO_COD_REQUISICAO	INTEGER	NN	Código da Requisição, único, maior que 0
FK	TIPO_ENTRADA_SAIDA_COD	INTEGER	NN	Código do tipo de entrada ou saida, maior que 0
FK	PRODUTO_COD_PRODUTO	INTEGER	NN	Código do Produto, maior que 0
	QUANTIDADE	FLOAT	NN	Quantidade, com 2 casas decimais
	DATA_MOV	DATE	NN	Data Movimentação, formato DD/MM/AAAA

CENTRO_CUSTO – Tabela armazena as informações dos Centro de Custos .				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD	INTEGER	NN	Código Centro de Custo, maior que 0, único, auto incremento
	NOME	VARCHAR(20)	NN	Nome do centro de custo

TIPO_ENTRADA_SAIDA – Tabela armazena os tipos de entrada e saída para uso nas movimentações.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD	INTEGER	NN	Código do tipo de entrada e saída, maior que 0, único, auto incremento
	TIPO	CHAR	NN	Tipo (E=entrada, S=saída)

ARMAZEM – Tabela contém informações dos armazéns onde cada produto se encontra.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD	INTEGER	NN	Código do Armazém, maior que 0, único, auto incremento.
	NOME	VARCHAR(20)	NN	Nome Armazém.

UNIDADE – Tabela armazena informações das Unidades de medida do produto.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD_UN	CHAR(2)	NN	Código de unidade medida (pc, m2, un, cx)
	NOME	VARCHAR(20)	NN	Nome das unidades de medida.

TIPO – Tabela que armazena o tipo de cada produto, ou seja, se ele é Matéria Prima, Produto Acabado entre outros.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD_TIPO	CHAR(2)	NN	Código do tipo produto (mp(Matéria prima), pa(Produto Acabado), pi(Produto Intermediário)).
	NOME	VARCHAR(20)	NN	Nome dos Tipos.

GRUPO – Tabela que armazena os grupos de cada produto, ou seja, se ele é Plástico, Borracha entre outros.				
Chave	**Atributo**	**Tipo**	**Valor Padrão**	**Comentário**
PK	COD_TIPO	INTEGER	NN	Código do grupo que o produto pertence, maior que 0
	NOME	VARCHAR(20)	NN	Nome do grupo.

APÊNDICE H – Artigo

Estudo de Modelagem Lógica de Sistemas da Área da Saúde, Industrial e Comercial, para Manipulação em um Repositório em Banco de Dados Objeto-Relacional

Tiago Rosa[1], Paracelso de Oliveira Caldas[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC

[2] Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC

tiago-rosa_@hotmail.com, poc@unesc.net

Abstract. *With regard to software development, logical modeling and data storage in a database object-relational, has a significant differential in the software market. To do so, creating a more agile way of modeling, enabling the reduction of deployment time and costs of the project, it is of great value to the acquirer. The study results show a difficulty in creating a prototype, being necessary to use an external tool, where developer systems use the information in the database and modeling will make from this information.*

Resumo. *No que se refere a desenvolvimento de software, a modelagem lógica de dados e a armazenagem em um banco de dados objeto-relacional, possui um diferencial significativo no mercado de softwares. Para tanto, a criação de uma forma de modelagem mais ágil, possibilitando a redução do tempo de implantação e custos do projeto, torna-se de grande valia para o adquirente. Os resultados do estudo evidenciam uma dificuldade na criação de um protótipo, sendo necessária a utilização de uma ferramenta externa, onde desenvolvedor de sistemas utilizará as informações existentes no banco de dados e fará as modelagens a partir destas informações.*

1. Introdução

A utilização da informática em constante evolução tem demonstrado ser o grande aliado nos processos de construção de conhecimento que envolve dados. Por sua vez o desenvolvimento de *software* vem sendo de crescente importância para sociedade. No entanto, deve-se levar em consideração os métodos usados nesta área de conhecimento, iniciando-se pelo desenvolvimento das habilidades de raciocínio lógico, atrelados à área programação, estruturas de dados e análises.

Com o intuito de melhorar a qualidade dos produtos de *software* e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de *Software* que pode ser auxiliada com as técnicas e metodologias, especificamente falando na engenharia de requisitos.

Diante do exposto, este trabalho analisa e identifica modelagens lógicas já existentes testadas e depuradas em Sistemas nas Áreas de Saúde, Industrial e Comercial, para Manipulação em um Repositório em Banco de Dados Objeto-relacional.

2. Engenharia de Software

Com o advento da evolução dos sistemas computadorizados, grande parte dos esforços e custos gerados, concentrava-se no desenvolvimento do *hardware* em razão das limitações encontradas na época. Posteriormente, com o crescimento em meio a necessidades, nasceu o termo "Engenharia de *Software*", e atualmente os preços dos equipamentos não acompanham a mesma tendência do desenvolvimento de *software*, o custo de um sistema informatizado, corresponde a uma percentagem cada vez maior.

Segundo Sommerville (2003, p. 6), "engenharia de *software* é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema".

Na engenharia de software, encontram-se vários processos de *software* definidos na literatura da Engenharia de *Software*. Dentre os vários processos existentes, existem as metodologias tradicionais, que são orientadas a documentação, e as metodologias ágeis, que procuram desenvolver *software* com o mínimo de documentação. Na sequência, apresenta-se alguns modelos conhecidos e utilizados no desenvolvimento de *software*s.

2.1 Modelo Clássico

O modelo clássico também chamado de Modelo em Cascata compreende em uma sistemática, sequencial ao desenvolvimento do *software*.

O modelo o cascata descreve um método de desenvolvimento linear e sequencial permitindo um controle departamental e gerencial, seguindo uma ordem estrita, sem qualquer sobreposição ou passos iterativos (PRESSMAN, 2006, 38-39).

No Ciclo de Vida Clássico utilizam-se conceitos de Engenharia de *Software*, a qual prevê atividade de verificação, validação e de controle de qualidade (PARREIRA JUNIOR, 2011, p. 22). Trata-se de um paradigma que utiliza um método sistemático e sequencial em que o resultado de uma fase se constitui na entrada de outra fase.

2.2 Prototipação

O modelo de desenvolvimento baseado em um protótipo relaciona-se a uma versão preliminar do sistema apresentado ao usuário, onde ele fornece informações ao desenvolvedor, para que sejam realizadas adaptações e implementações, durante o projeto e desenvolvimento.

Segundo Mazzola (2010, p. 1.10), "o objetivo da prototipação é um modelo de processo de desenvolvimento que busca contornar limitações existentes no modelo cascata, eliminando a política de congelamento dos requisitos antes do projeto ou codificação".

De acordo com Oliveira L., (2009, p. 6), no modelo prototipagem (pura), "o desenvolvedor interage diretamente com o usuário, escutando seus pedidos e desenvolvendo, imediatamente, um protótipo do produto desejado".

Com este modelo, o usuário utiliza o protótipo e fornece ao desenvolvedor outras novas informações que o levam a gerar atualizações, adaptações e implementações no *software*, durante o de projeto e desenvolvimento.

2.3 Modelo Espiral

No modelo espiral mostra-se que as diferentes atividades são repetidas até uma decisão ser tomada e o documento de especificação de requisitos ser aceito. Compreende-se com este modelo, como sendo de uma abordagem "evolucionária" à engenharia de *software*, capacitando o desenvolvedor e o cliente a entender e reagir aos riscos em cada fase evolutiva.

O modelo espiral exige uma consideração direta dos riscos técnicos em todas as etapas do projeto e, se adequadamente aplicado, deve reduzir os riscos antes que eles se tornem problemáticos (Pressman, 2006, p. 45).

Observou-se que o Modelo de Desenvolvimento em Espiral, surgiu com o objetivo de resolver os problemas dos projetos que seguem o fluxo que o modelo Cascata propõe. No entanto, no início do projeto torna-se difícil para o cliente especificar os requisitos.

2.4 Desenvolvimento Iterativo e Incremental

Necessita-se que um processo de desenvolvimento de *software* seja iterativo, isto é, ter várias iterações no tempo. Como também, necessita que seja incremental, isto é, gerar novas versões incrementadas a cada release.

Para Silva, Souza e Dantas (2006, p. 18), "o modelo iterativo e incremental é uma alternativa para solução de problemas enfrentados no modelo cascata".

Segundo Bezerra (2003), "um processo de desenvolvimento segundo essa abordagem divide o desenvolvimento de um produto de *software* em ciclos".

O modelo em questão possibilita desenvolvimento do *software* em módulos de acordo com as necessidades e características do projeto. A abordagem incremental e iterativa somente torna-se possível existindo um mecanismo para dividir os requisitos do sistema em partes, e cada parte alocada a um ciclo de desenvolvimento.

2.5 Metodologia Ágil

A evolução constante no desenvolvimento de *softwares* tornou-se referência quando um grupo de especialistas estabeleceu alguns quesitos. Desde então, aponta-se as metodologias ágeis como uma alternativa às abordagens tradicionais no desenvolvimento de *software*.

Segundo Banki e Tanaka (2008, p. 22), "as metodologias ágeis, como a Extreme Programming (XP) e o Scrum, entre outras, têm despertado atenção crescente do mercado".

A Programação Extrema propõe um conjunto de valores, princípios e práticas, que visam garantir o sucesso no desenvolvimento de *software*, em face a requisitos vagos e que mudam constantemente (SATO, 2007, p. 3).

Esse movimento, baseado no ciclo de desenvolvimento incremental e iterativo, foca-se na colaboração do cliente, no valor dos indivíduos e na adaptação às mudanças.

3. Engenharia de Requisitos

A engenharia de requisitos trata-se de uma importante parte da engenharia de *software*, onde as empresas investem no aprimoramento técnico, como também o processo onde identifica-se os serviços que o sistema deve oferecer e as restrições que deve respeitar, estabelecendo uma base sólida para o projeto e construção. Sem a engenharia, o resultado do *software* tem a probabilidade de não satisfazer às necessidades dos clientes.

Conforme Aurum e Wohlin (2005 apud Falbo, 2011, p. 1), "engenharia de requisitos é o processo pelo qual os requisitos de um produto de *software* são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de vida do *software*".

A Engenharia de Requisitos tem se tornado cada vez mais necessária para resolver os problemas encontrados nas organizações com relação à definição de sistemas (GASTALDO; MIDORIKAWA, 2004, p. 1).

3.1 Análise de Requisitos

Objetiva-se com a análise de requisitos, o tratamento do processo de definição dos requisitos do *software* a ser desenvolvido. Com isso, elabora-se as atividades criteriosamente, compreendendo o que espera-se do aplicativo.

Conforme Oliveira, L. (2009, p. 10), "esta fase visa identificar o tipo de serviço de processamento de dados a ser executado, os objetivos a serrem alcançados, recursos e prazos necessários para a execução do projeto".

Sobre o assunto, Sommerville (2003, p. 82), faz uma classificação de requisitos segundo seus níveis de descrição, sendo requisitos de usuário as "declarações, em linguagem natural e também em diagramas sobre as funções que o sistema deve fornecer e as restrições sobre as quais deve operar" e, requisitos de sistemas, como as descrições detalhadas das funções e restrições do sistema.

A partir dessa classificação o autor define os requisitos funcionais e não-funcionais. Onde os requisitos funcionais referem-se às declarações de funções que o sistema deve fornecer e como o sistema deve reagir a entradas específicas, como também deve se comportar em determinadas situações e os requisitos não-funcionais referem-se às restrições sobre os serviços ou as funções oferecidos pelo sistema.

3.2 Modelagem Lógica de Bando de Dados

Inicia-se o modelo lógico a partir do modelo conceitual, considerando as abordagens, Relacional, Hierárquica e Rede. Um modelo lógico trata-se de uma definição de um banco de dados no grau de abstração visto pelo usuário do SGBD.

Assim, o modelo lógico torna-se dependente do tipo particular de SGBD que está sendo usado (HEUSER, 2001, p. 17).

Segundo Berwanger (2009, p. 14), "um modelo enfatiza um conjunto de características da realidade, que corresponde à dimensão do modelo".

Dá-se com o modelo uma visão do projeto antes de sua execução, sendo ela, externa do projeto, ajudando a reconhecer e até a corrigir erros antes mesmo desses acontecerem, tornando-se um projeto confiável.

3.3 Modelagens de Sistemas

A modelagem de sistemas inicia-se por uma distinção que associa a maneira de perceber os fenômenos ao modo de pensamento que predomina em cada indivíduo, na observação de um determinado fenômeno, num momento específico do tempo.

Para Silva, (2005), em seu artigo publicado na internet, "os estudos destes sistemas podem dar-se sob diferentes formas de abordagem".

Neste contexto a modelagem de *software* consiste em auxiliar na organização das informações, estabelecendo um escopo para o projeto que descreva a real utilidade perante o *software*, levando em consideração as análises de requisitos o *software* terá sua validação em seu conjunto.

3.4 UML – Linguagem de Modelagem Unificada

A UML, Linguagem de Modelagem Unificada (Unified Modeling Language), surgiu para oferecer uma modelagem de forma a abranger todas as visões de um sistema de *software*, possibilitando uma maior interação entre analista, projetistas, programadores e demais profissionais envolvidos no desenvolvimento de um sistema.

UML ou Unified Modeling Language é uma unificação dos métodos OMT, Booch e OOSE que está sendo submetido a OMG para a padronização (MAZZOLA, 2009, p. 9.5).

Conforme Berwanger (2009, p. 16), "é uma linguagem padrão para a elaboração da estrutura de projetos de *software*".

A UML fornece uma maneira para capturar e discutir exigências (Diagramas de Caso do Uso). Encontra-se diagramas para capturar as partes do sistema de *software* que realizam determinadas exigências (Diagramas de Colaboração). Diagramas para capturar exatamente como aquelas partes do sistema realizam suas exigências (Diagramas de Sequência e de Gráfico de Estado). Finalmente, diagramas para mostrar como tudo se unem para juntos serem executados (Diagramas de Componentes e de Distribuição). Encontram-se diversos tipos, alguns com finalidades específicas e alguns com usos mais genéricos.

3.5 Diagramas

Entende-se por diagramas o estudo gráfico dos esforços que retratam os valores dos esforços simples ao longo da estrutura, permitindo a visualização das variações desses esforços de uma seção para outra.

Segundo Ferrari (2006, p. 47), "a UML modela um sistema e sua interação com o usuário por meio de diagramas padronizados".

Os diagramas UML esboçam todas as visões de um projeto de sistema de *software*. São treze diagramas apresentados na literatura, no entanto esta abordagem enfatizará somente os diagramas de caso de uso (Use Case), diagrama de sequência, diagrama de colaboração e diagrama de atividades.

Um Diagrama de Caso de Uso modela a funcionalidade fornecida pelo sistema. Segundo Savi (2009, p. 84), diagrama de caso de uso mostra como o sistema vai interagir com os usuários (pessoas ou outros sistemas). Desta forma diagramas de caso de uso descrevem as interações e funcionalidades entre as categorias de usuários e o sistema.

Um Diagrama de Sequência modela a sequência funcional do sistema de *software*, dando uma sequência temporal aos itens presentes nele. Tradicionalmente os diagramas de sequência mostram os tipos de objetos envolvidos no caso de uso, as mensagens que trocam entre si e qualquer valor retornado associado com a mensagem (JUNQUEIRA, 2006, p. 100).

Um Diagrama de Colaboração, segundo Savi (2009, p. 84), "mostra as interações entre vários componentes do sistema". Neste contexto, pode-se utilizar o diagrama de colaboração para substituir o diagrama de sequência.

Observa-se que ambos os diagramas modelam interações entre objetos para uma tarefa específica, mas enquanto o diagrama de sequência enfatiza o arranjo na sequência de interações sobre o tempo, o diagrama de colaboração enfatiza a organização estrutural dos objetos que enviam e recebem mensagem, desta forma, modela a forma das interações utilizam a estrutura dos objetos e a participação em seus relacionamentos.

Um Diagrama de Atividades segundo Savi (2009, p. 84), "mostra como um subsistema ou um objeto realiza uma operação, também conhecido como diagrama de fluxo de dados".

Conforme Junqueira (2006, p. 101), "o diagrama de atividades modela a sequência de atividades em um processo, bem como seus elementos ativos e passivos".

Documentam-se os diagramas de atividades com uma breve descrição da atividade e uma indicação da ação que ocorre (ou das ações que ocorrem) durante um processo. O

diagrama de atividades mostra um conjunto de atividades, o fluxo de cada atividade para com outra, e os objetos que realizam ou sofrem ações.

4. Banco de Dados

Os sistemas de gerenciamento de banco de dados (SGBD) surgiram no início da década de 70 com o objetivo de facilitar a programação de aplicações de banco de dados.

Conforme Alexandruk (2011, p. 2), "um banco de dados pode ser definido como uma coleção de dados integrados, que tem por objetivo atender a uma comunidade de usuários".

Como também um conjunto de dados persistentes e manipuláveis que obedecem a um padrão de armazenamento. Por "dados" pode-se compreender como fatos conhecidos que podem ser armazenados e que possuem um significado implícito.

4.1 Sistema de Gerenciamento de Banco de Dados

Ao longo do tempo, os SGBD's evoluíram desses sistemas de arquivos de armazenamento em disco, criando-se novas estruturas de dados com o objetivo de armazenar informações.

Segundo Alexandruk (2011, p. 2), "o Sistema de Gerenciamento de Banco de Dados é o *software* que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados".

Permite-se com um SGBD, que os usuários criem e manipulem bancos de dados de propósito gerais. Desta forma, o conjunto formado por um banco de dados, mais as aplicações que manipulam o mesmo é chamado de "Sistema de Banco de Dados".

4.2 Arquitetura e Modelagem de Banco de Dados

As primeiras arquiteturas usavam mainframes para executar o processamento principal, bem como o de todas as funções do sistema, incluindo os programas aplicativos de interface com o usuário, como também a funcionalidade do SGBD. Com esta arquitetura, os usuários acessavam somente pelos terminais, os quais não possuíam processamento e apenas permitia visualização. Desta forma, os processamentos de todos os dados eram feitos remotamente, conectados por redes de comunicação.

De acordo com Alexandruk (2011, p. 2), "o modelo de dados refere-se à descrição formal da estrutura de um banco de dados".

Desta forma, para construir um modelo de dados, utiliza-se uma linguagem de modelagem de dados. Estas linguagens de modelagem de dados podem ser classificadas de acordo com a forma de apresentar modelos, em linguagens textuais ou linguagens gráficas.

4.3 Modelo de Banco de Dados

Os bancos de dados apareceram no final dos anos 60, numa época em que a necessidade de um sistema de gestão da informação flexível passava a ser necessária diante da evolução constante, no entanto já no fim dos anos 90, as bases relacionais relacionavam-se aos bancos de dados mais comuns.

De acordo com Laundon & Laundon (1999 apud Araújo, 2003, p. 8), "existem três modelos de projeto de lógico de banco de dados: o modelo hierárquico, o modelo em rede e o modelo relacional".

O modelo hierárquico trata-se do primeiro modelo de SGBD, onde classifica-se os dados de acordo em forma hierárquica descendente, utilizando-se de apontadores entre os diferentes registros. Conforme Langa (2007), "este modelo utiliza árvores para a

representação lógica dos dados, composta de elementos chamados nós, onde o nível mais alto da árvore denomina-se raiz".

Semelhante ao o modelo hierárquico, o modelo em rede utiliza apontadores para os registros. Desta forma a estrutura não necessariamente utiliza do formato em árvore invertida e descendente. Neste modelo as entidades se representam como nós e suas relações são as linhas que os unem, desta forma nesta estrutura qualquer componente pode se relacionar com qualquer outro (LANGA, 2007).

No modelo relacional, registram-se os dados em duas dimensões, isto é, linhas e colunas. Para Araújo (2003, p. 9), "o modelo de Banco de Dados Relacional é a mais recente estrutura de banco de dados. Representa todos os dados do banco de dados em tabelas simples bidimensionais denominadas relações".

O modelo relacional trata-se do mais moderno, bem como possui a mais adequada metodologia, o que o deixa em situação privilegiada sobre os demais, no desenvolvimento de projeto lógico de banco de dados.

4.4 Sistema de Banco de Dados Orientados a Objetos

Segundo Silveira et al (2006, p. 2), "os sistemas de banco de dados orientado a objetos, cuja sigla em inglês é OODBMS, surgiram na década de 80, fruto de pesquisas sobre o armazenamento de dados estruturados sob a forma de grafos".

Conforme Silveira, et al (2006, p. 7), "o SGBD Objeto-Relacional (SGBDOR) emprega um modelo que coloca a orientação a objetos em tabelas, unindo dois paradigmas em um só". Essa tecnologia permite que as aplicações persistam os dados de forma transparente, em qualquer tipo de banco. Para a aplicação, trata-se como se existisse um banco de dados orientado a objeto servindo de repositório.

5. Repositório de Armazenamento de Modelagem Lógica de Sistemas

Tendo como finalidade a redução do tempo e custos na implantação de *softwares*, este estudo tem como objetivo a verificação de três tipos de sistemas atuantes no mercado de *software*, dando enfoque principal ao sistema para indústria.

Em análise à relevância do tema, o estudo apresenta a identificação e modelagem lógica de aplicações já existentes, aprovadas e consolidadas no mercado, utilizando métodos e padrões homologados para o armazenamento e manipulação em um banco de dados objeto-relacional.

O protótipo permite que o desenvolvedor do *software,* execute as operações a partir de um banco de dados existente, uma vez que parte das informações e dos dados já estejam estruturados para utilização nos três tipos de sistemas, reduzindo assim o tempo de implantação e custos do projeto. Desta forma, utilizou se da ferramenta *Astah Community*, para a modelagem de dados, o *Microsoft SQL Server Express* como banco de dados e a linguagem de programação *Visual FoxPro*.

Iniciada a implementação do protótipo, seguido dos testes necessários, implementado a funcionalidade de manipulação de modelos existentes (criado pela ferramenta Astah), assim dando a possibilidade de gravar num banco de dados.

O *Astah Community* ferramenta utilizada para modelagens de dados foi utilizado como ferramenta externa para modelagem dos diagramas. Mesmo com limitações de uma versão grátis, a Astah Community trata-se de uma ferramenta adequada para modelagem UML.

5.1 Protótipo Desenvolvido

O protótipo desenvolvido serve para manipulação de um repositório de modelagem lógica de sistemas desenvolvidos, testados e depurados, para serem usados na manutenção de sistemas em uso ou para a construção de novos. Para o desenvolvimento do protótipo, foram utilizadas as ferramentas *Microsoft SQL Server Express* para armazenagem em banco de dados e *Visual FoxPro* como linguagem de programação, as quais servem de suporte para armazenagem em banco de dados e linguagem de programação.

5.2 Repositório de Modelagem lógica

O sistema tem o intuído de manipular, ou seja, importar uma modelagem existente, para um repositório (Banco de dados).

O sistema é constituído por duas telas, onde na tela principal conforme mostrado a baixo tem-se a opção de cadastro de um novo módulo, conforme o seu segmento. Na ilustração abaixo, visualiza-se 3 módulos cadastrados para o segmento Industrial (Estoque, Suprimento, Comercial). A Figura 2 possibilita ao usuário a escolha do diagrama desejado.

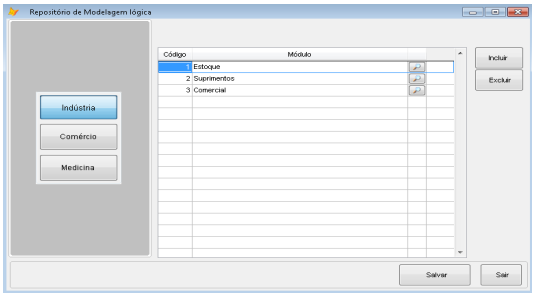


Figura 1. Repositório de Modelagem Lógica.

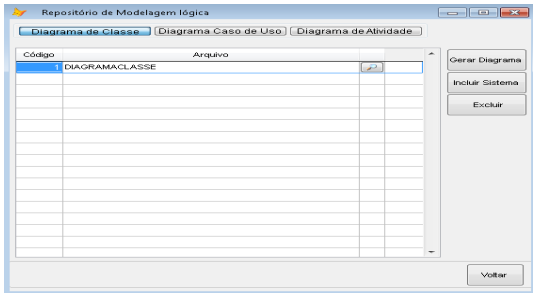


Figura 2. Opção da Seleção de Diagramas.

Para implementação do projeto foi utilizada Modelagem UML a qual combinando-a com um modelo de processo resulta na criação de aplicativos bem-sucedidos.

6. Resultados

Os resultados obtidos através da criação deste repositório referem-se à facilidade dos engenheiros de *software* e clientes terem acesso ao acervo, podendo fazer consultas de modelos já existentes. Desta forma, esta consulta auxiliará na implantação do novo sistema, reduzindo o tempo real de uma implantação e também diminuindo os custos do projeto.

No protótipo foram catalogadas no repositório a modelagem dos sistemas nas áreas de: Saúde, Comércio e Indústria que podem ser manipulados e usados na manutenção de sistemas em funcionamento ou na criação de novos sistemas.

Para as funcionalidades foram usados Diagramas de Casos de Uso, apresentando as funcionalidades do sistema Industrial.

A modelagem do banco de dados foi em UML pelo diagrama de classe entidade, além de servir também para especificação do dicionário de dados. Para apresentar o fluxo e as ações, (caminho, cálculos, decisões etc) das atividades do sistema de Industrial, foi representado através do diagrama da atividade de cadastrar fornecedores.

7. Considerações Finais

Com este trabalho, foi possível contribuir para a evolução das técnicas de modelagem, em especial o desenvolvimento de um protótipo para manipulação destas modelagens, de uma forma estruturada em uma base de dados (repositório).

Através do estudo, identificaram-se as modelagens lógicas no desenvolvimento dos *softwares* estudados, possibilitando catalogar suas informações utilizando a linguagem de modelagem UML. E por fim, fornecer estas informações por meio de um protótipo, para futuras consultas.

Houve dificuldade para criar o protótipo no que se refere a apresentação dos modelos lógicos, ou seja, os diversos diagramas para visualização do sistema. Para resolver este problema foi utilizada a ferramenta Astah a qual manipula os modelos armazenados no repositório pelo protótipo.

Conclui-se que o desenvolvimento de *software* ganha um aliado no aumento da qualidade, redução de tempo no desenvolvimento, consequentemente diminuição de custos, devido a um repositório de modelos lógicos de aplicações, onde as fases de levantamento, análise, modelagem e especificação de requisitos, dos modelos prontos obtidos no repositório, são eliminadas.

Referências

ALEXANDRUK, Marcos. **Modelagem de banco de dados.** 2011. Disponível em: <www. Unilivros.com.br/pdf/dbmod.pdf>. Acesso em: 22 mai. 2011.

ARAÚJO, Dinaldo do Nascimento. Modelo lógico de banco de dados proposto para uma contabilidade de partida múltipla. **Adcontar**, Belém, v. 4, n. 2, p. 27-44, nov. 2003.

BANKI, André Luis; TANAKA, Sérgio Akio. Metodologias ágeis: uma visão prática. **Engenharia de *Software* Magazine**, Rio de Janeiro, ano 1, ed. 4, p. 22-29, 2008.

BERWANGER, Elielder. Apostila Engenharia de Software. 2009. Disponível em: <http: // www.fag.edu.br/professores/elielder/materias/.../apostila_eng_software.pdf >. Acesso em: 20 mai. 2011.

BEZERRA, Eduardo. **Princípios de analise e projeto de sistemas com UML**. Rio de Janeiro: Campus, 2003. 286 p. Disponível em: < http://wiki.sj.ifsc.edu.br/wiki/index .php/Ciclo_de_Vida_Iterativo_e_Incremental>. Acesso em: 20 mai. 2011.

FALBO, Ricardo de Almeida. **Engenharia de Requisitos**. Espírito Santo, 2011. UFES - Universidade Federal do Espírito Santo. Disponível em: < www.inf.ufes.br/~falbo/ files/Notas_Aula_Engenharia_Requisitos.pdf >. Acesso em: 14 mai. 2011.

FERRARI, Thales Martins. **Sistema de *Software* para auxílio dos docentes da Unifei**. 2006. 138 f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Itajubá, Minas Gerais.

GASTALDO, Denise Lazzeri; MIDORIKAWA, Edson Toshimi. Processo de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho: um estudo de caso. In: INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING., Piracicaba, SP, 2004. **WER03 proceedings.**. Rio de Janeiro: PUC, p. 302-328. 2004.

HEUSER, Carlos Alberto. **Projeto de banco de dados.** 4. ed Porto Alegre: Sagra Luzzatto, 2001. 204 p.

JUNQUEIRA, Fabrício. **Modelagem e simulação distribuídas de sistemas produtivos**. 2006. 222 f. Tese (Doutorado em Engenharia Mecatrônica e de Sistemas Mecânicos). Escola Politécnica da Universidade de São Paulo, São Paulo.

LANGA, Sara Alvarez. **Modelo de bancos de dados**. 2007. Disponível em: < http:// www.criarweb.com/artigos/modelos-banco-dados.html >. Acesso em: 25 mai. 2011.

MAZZOLA, Vitório Bruno. **Engenharia de *Software***: Conceitos Básicos. Florianópolis, 2010. INE/CTC/UFSC. Disponível em: < http://algol.dcc.ufla.br/~monserrat/icc/ Introducao_ES.pdf >. Acesso em 15 mai. 2011.

OLIVEIRA, Luiz Ângelo. **Ciclo de vida de sistemas de informação**. São Paulo, 2009. Disponível em: < http://www.eteavare.com.br/arquivos/43_37.pdf >. Acesso em: 14 mai. 2011.

PARREIRA JUNIOR, Walteno Martins. **Engenharia de *Software***. Universidade do Estado de Minas Gerais. 2011. 108 f. Apostila - Fundação Educacional de Ituiutaba. Curso de Engenharia da Computação e Engenharia de *Software*. Disponível em: < http://www. waltenomartins.com.br/ap_es_v1.pdf>. Acesso em: 13 mai. 2011.

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

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de *software***. 2007. 139 f. Dissertação (Mestrado em Ciências da Computação) – Universidade de São Paulo, São Paulo.

SAVI, Antonio Francisco. **Modelo de sistema para gerenciamento de conhecimentos explícitos em abordagens de DFA (Design For Assembly).** Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

SILVA, Cristiano Campos; SOUZA, Kleyton Farias de; DANTAS, Samuel Dias. **Metodes: Metodologia de desenvolvimento de *software***. 2006. 159 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Faculdade Cenecista de Brasília. Distrito Federal.

SILVA, Luís César. **Simulação de Processos.** Universidade Federal do Espírito Santo. 2005. Disponível em: < http://www.agais.com/simula.htm>. Acesso em: 10 mai. 2011.

SILVEIRA, Daniel Gomes; ALVES, Filipe Ferreira; FERREIRA, Leandro Soriano; SOUZA, Rodrigo Rocha Gomes e. Banco de dados orientados a objetos-relacionais. 2006. Departamento de Ciências da Computação da Universidade Federal da Bahia. Disponível em:<http://im.ufba.br/pub/MATA60/SemestreLetivo20062/Bancos_de_dados_OO_e_OR. pdf >. Acesso em: 28 mai. 2011.

SOMMERVILLE, Ian. **Engenharia de *software*/** Ian Sommerville. São Paulo: Addison-Wesley, 2003. 592 p.